ChatDB: A SQL like chat database software based on console interaction

Liqiang Deng

Tech stack and databse select

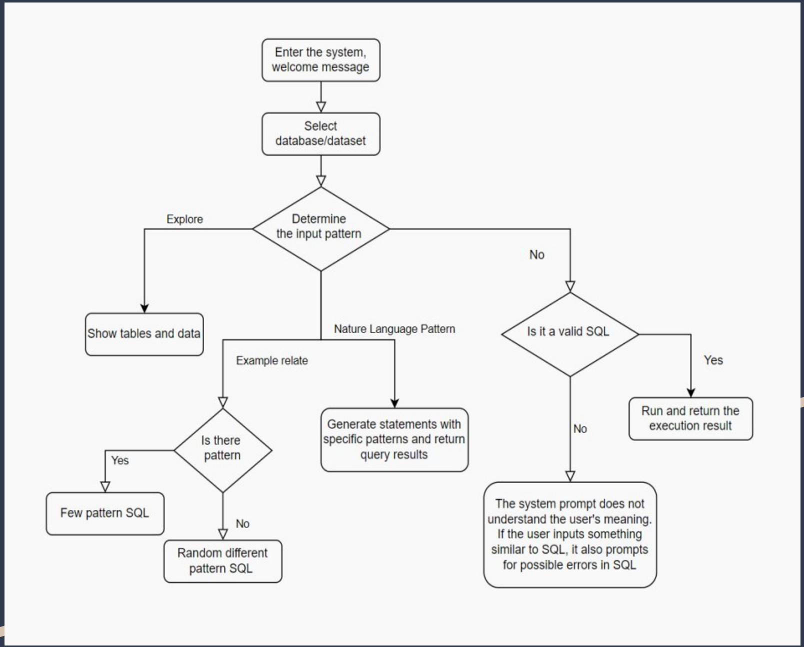
Tech stack:

I have been using Python to program, using mysql-connector-python to connect to the database, using re for matching patterns in SQL example and user nature language parsing, using csv package to handle raw dataset.

Database:

As I am a single member team, I have decided to use MySQL as the only support.

Main Process


Enter the system, welcome message
Select database/dataset
Determine the input pattern
Explore
Show tables and data
Nature Language Pattern
Example relate
Is there pattern
Yes
Few pattern SQL
No
Random different pattern SQL
Generate statements with specific patterns and return query results
No
Is it a valid SQL
Yes
Run and return the execution result
No
The system prompt does not understand the user's meaning. If the user inputs something similar to SQL, it also prompts for possible errors in SQL

After the user starts running the system's main.py program, the system basically follows the following process to determine the user's input and return the result. User can exit the program through exit command.

User can type ‘help’ to see guildline.

If the user's input includes 'upload', you will be able to select a pre filled CSV file to upload the data to the system.

If the user's input includes 'show table', you will be able to view the existing data tables and table structures in the database.

If the user's input includes 'example', you will be able to get some example SQL from database.

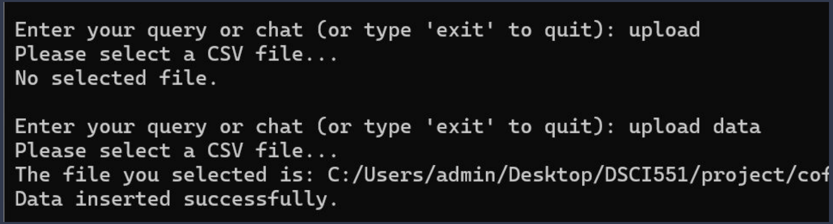
And you can include some keyword such as 'group by', 'sum' to get specific SQL example.

IF the user inputs a valid SQL statement, the system will run directly and return the result to you.

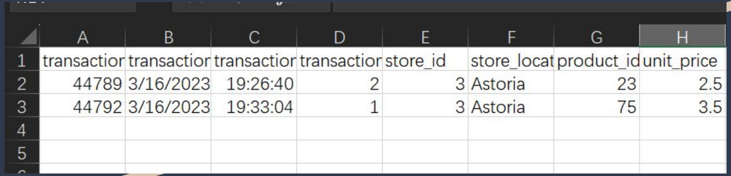
If the above situations are not met, the system will attempt to understand the user's input and obtain corresponding SQL query statements and explanations.

If the input is incomprehensible to the system, the system will prompt the user to re-enter.

Upload & Show



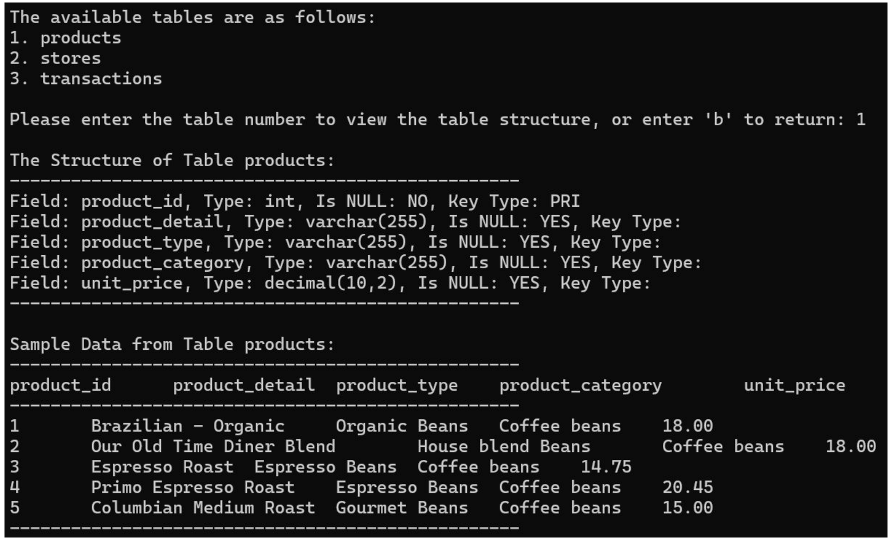
```
Enter your query or chat (or type 'exit' to quit): upload
Please select a CSV file...
No selected file.

Enter your query or chat (or type 'exit' to quit): upload data
Please select a CSV file...
The file you selected is: C:/Users/admin/Desktop/DSCI551/project/cof
Data inserted successfully.
```



	A	B	C	D	E	F	G	H
1	transactior	transactior	transactior	transactior	store_id	store_locat	product_id	unit_price
2	44789	3/16/2023	19:26:40	2	3	Astoria	23	2.5
3	44792	3/16/2023	19:33:04	1	3	Astoria	75	3.5
4								
5								

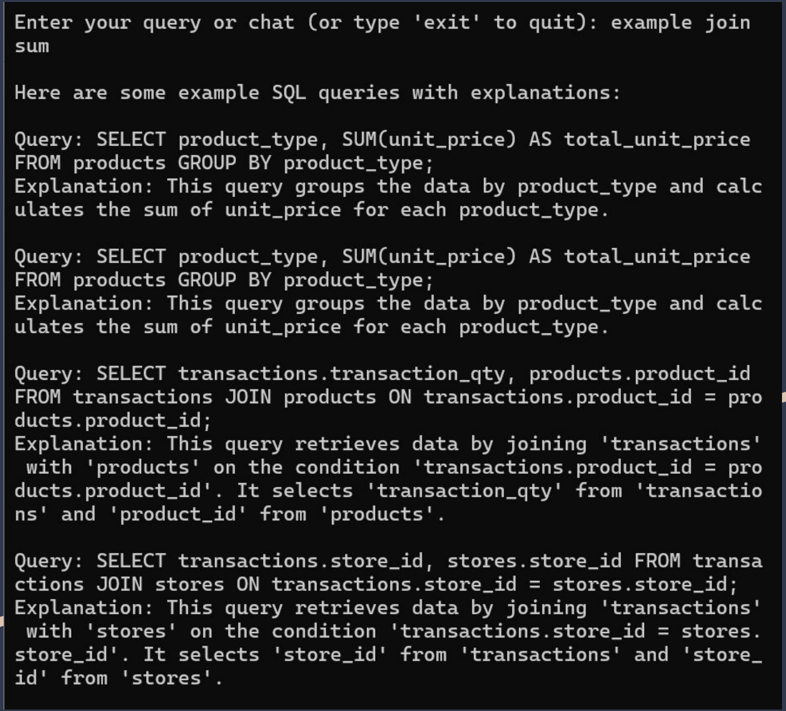
Users can activate interactive reading of legitimate CSV files, validate and upload them to the system database by entering statements containing the keyword 'upload'.

Users can input statements containing 'show table' to display all tables in the database, as well as enter corresponding numbers to view the table structure and some sample data.



```
The available tables are as follows:
1. products
2. stores
3. transactions

Please enter the table number to view the table structure, or enter 'b' to return: 1

The Structure of Table products:
--------------------------------------------------
Field: product_id, Type: int, Is NULL: NO, Key Type: PRI
Field: product_detail, Type: varchar(255), Is NULL: YES, Key Type:
Field: product_type, Type: varchar(255), Is NULL: YES, Key Type:
Field: product_category, Type: varchar(255), Is NULL: YES, Key Type:
Field: unit_price, Type: decimal(10,2), Is NULL: YES, Key Type:
--------------------------------------------------

Sample Data from Table products:
--------------------------------------------------
product_id      product_detail  product_type    product_category        unit_price
--------------------------------------------------
1       Brazilian - Organic     Organic Beans   Coffee beans    18.00
2       Our Old Time Diner Blend        House blend Beans       Coffee beans    18.00
3       Espresso Roast  Espresso Beans  Coffee beans    14.75
4       Primo Espresso Roast    Espresso Beans  Coffee beans    20.45
5       Columbian Medium Roast  Gourmet Beans   Coffee beans    15.00
--------------------------------------------------
```

Example SQL



```
Enter your query or chat (or type 'exit' to quit): example join
sum

Here are some example SQL queries with explanations:

Query: SELECT product_type, SUM(unit_price) AS total_unit_price
FROM products GROUP BY product_type;
Explanation: This query groups the data by product_type and calc
ulates the sum of unit_price for each product_type.

Query: SELECT product_type, SUM(unit_price) AS total_unit_price
FROM products GROUP BY product_type;
Explanation: This query groups the data by product_type and calc
ulates the sum of unit_price for each product_type.

Query: SELECT transactions.transaction_qty, products.product_id
FROM transactions JOIN products ON transactions.product_id = pro
ducts.product_id;
Explanation: This query retrieves data by joining 'transactions'
 with 'products' on the condition 'transactions.product_id = pro
ducts.product_id'. It selects 'transaction_qty' from 'transactio
ns' and 'product_id' from 'products'.

Query: SELECT transactions.store_id, stores.store_id FROM transa
ctions JOIN stores ON transactions.store_id = stores.store_id;
Explanation: This query retrieves data by joining 'transactions'
 with 'stores' on the condition 'transactions.store_id = stores.
store_id'. It selects 'store_id' from 'transactions' and 'store_
id' from 'stores'.
```

Users can input statements containing examples to command the system to randomly generate executable SQL statements. By adding keywords, the system can generate specific SQL query statements and corresponding explanations.

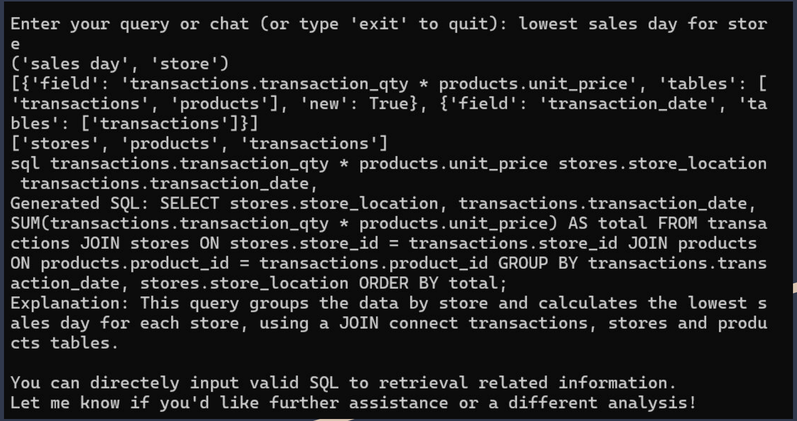
Now system support: 'group by', 'having', 'order by', 'where', 'limit', 'avg', 'sum', 'join'.

Impeachment:

Firstly, extract the user's input and check the keywords in it. And there is a dictionary Table_Fields that stores the types and ranges of values for all fields in all tables.

Then, based on each keyword, randomly select fields and values within the range to fill in template to generate two example SQL queries for each type.

Ask Question in NL



```
Enter your query or chat (or type 'exit' to quit): lowest sales day for stor
e
('sales day', 'store')
[{'field': 'transactions.transaction_qty * products.unit_price', 'tables': [
'transactions', 'products'], 'new': True}, {'field': 'transaction_date', 'ta
bles': ['transactions']}]
['stores', 'products', 'transactions']
sql transactions.transaction_qty * products.unit_price stores.store_location
 transactions.transaction_date,
Generated SQL: SELECT stores.store_location, transactions.transaction_date,
SUM(transactions.transaction_qty * products.unit_price) AS total FROM transa
ctions JOIN stores ON stores.store_id = transactions.store_id JOIN products
ON products.product_id = transactions.product_id GROUP BY transactions.trans
action_date, stores.store_location ORDER BY total;
Explanation: This query groups the data by store and calculates the lowest s
ales day for each store, using a JOIN connect transactions, stores and produ
cts tables.

You can directely input valid SQL to retrieval related information.
Let me know if you'd like further assistance or a different analysis!
```

The system will match specific patterns based on user input to understand the user's meaning, and then generate corresponding SQL statements and explanations.

Impeachment:

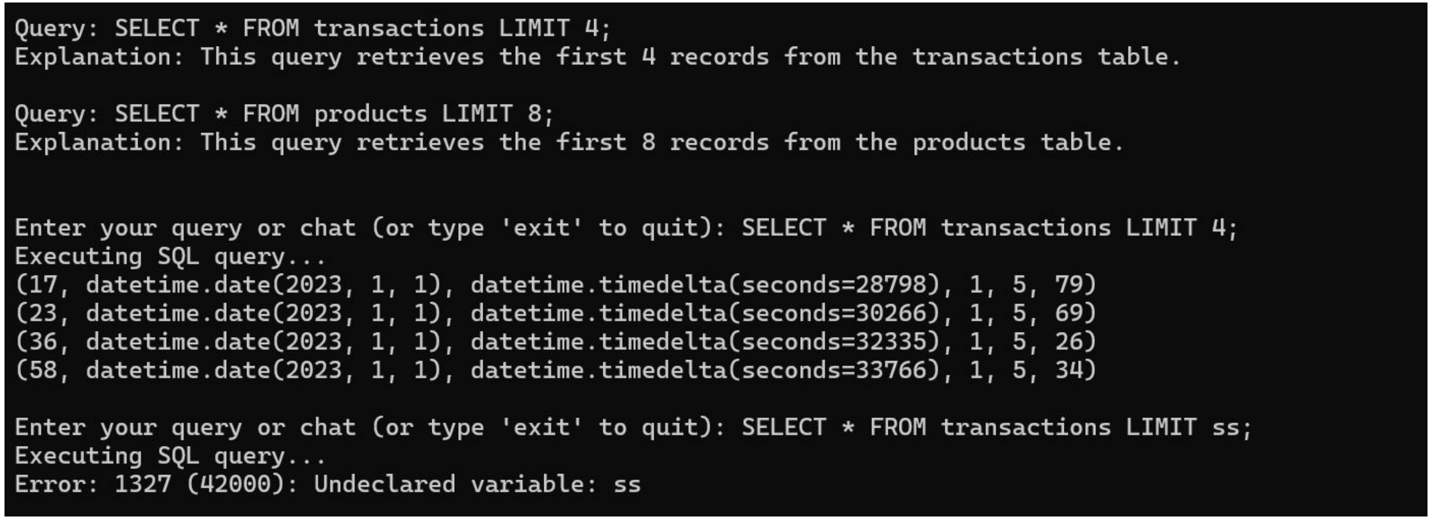
Multiple patterns are pre-set, and then one of them is matched through regular expression matching.

After matching, we divide the user input into several parts, match fields from them, and then obtain corresponding tables. If there are more than two tables, the system needs to use join to generate connection conditions. Otherwise, we can directly use fields to fill in the template and generate SQL queries.

Example：find total unit price by product, count transaction by date, count transaction by store, average price by product, max price by product, min price by product, top 5 price by product, filter product category where coffee, highest sales day for store, lowest sales day for product.

Directly run SQL queries

Users can directly input valid SQL statements into the system to run and obtain results. If SQL is illegal, you will receive a system prompt.



```
Query: SELECT * FROM transactions LIMIT 4;
Explanation: This query retrieves the first 4 records from the transactions table.

Query: SELECT * FROM products LIMIT 8;
Explanation: This query retrieves the first 8 records from the products table.


Enter your query or chat (or type 'exit' to quit): SELECT * FROM transactions LIMIT 4;
Executing SQL query...
(17, datetime.date(2023, 1, 1), datetime.timedelta(seconds=28798), 1, 5, 79)
(23, datetime.date(2023, 1, 1), datetime.timedelta(seconds=30266), 1, 5, 69)
(36, datetime.date(2023, 1, 1), datetime.timedelta(seconds=32335), 1, 5, 26)
(58, datetime.date(2023, 1, 1), datetime.timedelta(seconds=33766), 1, 5, 34)

Enter your query or chat (or type 'exit' to quit): SELECT * FROM transactions LIMIT ss;
Executing SQL query...
Error: 1327 (42000): Undeclared variable: ss
```

Additional & Future work

1. Add more language templates for matching pattern.
2. Support more filtering conditions, ‘>’, ‘>=’, ‘<>’, ‘<’, ‘<=’.
3. Support users to modify, delete data, and modify table structures.
4. …

Thank you!