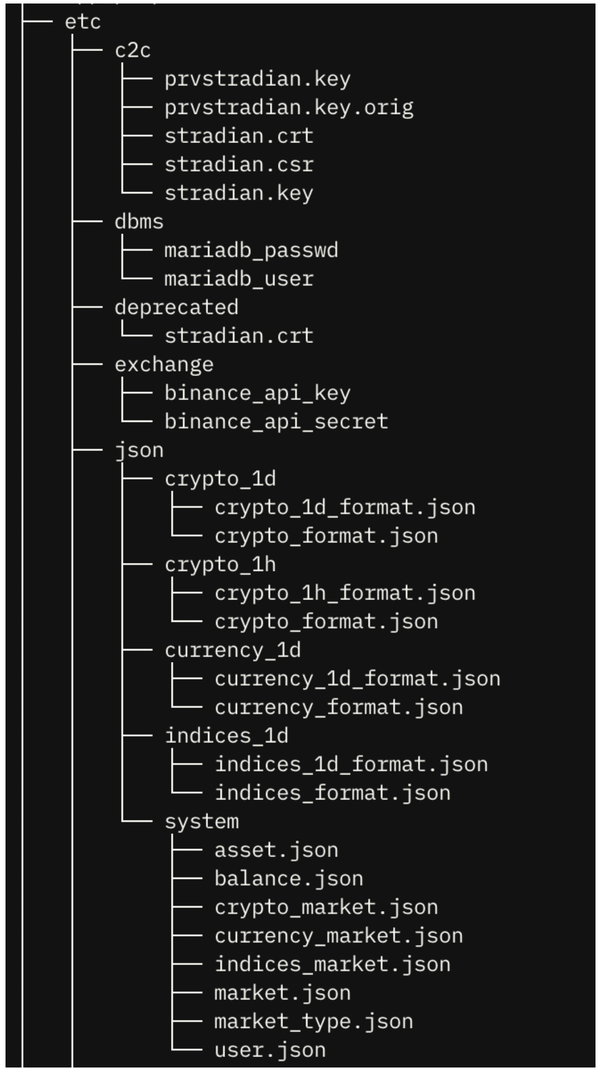
Project Demo Presentation

ChatDB-94

5023184813 Jeong Hoon Choi

Table of Contents

About the Project



```
├── etc
│   ├── c2c
│   │   ├── prvstradian.key
│   │   ├── prvstradian.key.orig
│   │   ├── stradian.crt
│   │   ├── stradian.csr
│   │   └── stradian.key
│   ├── dbms
│   │   ├── mariadb_passwd
│   │   └── mariadb_user
│   ├── deprecated
│   │   └── stradian.crt
│   ├── exchange
│   │   ├── binance_api_key
│   │   └── binance_api_secret
│   ├── json
│   │   ├── crypto_1d
│   │   │   ├── crypto_1d_format.json
│   │   │   └── crypto_format.json
│   │   ├── crypto_1h
│   │   │   ├── crypto_1h_format.json
│   │   │   └── crypto_format.json
│   │   ├── currency_1d
│   │   │   ├── currency_1d_format.json
│   │   │   └── currency_format.json
│   │   ├── indices_1d
│   │   │   ├── indices_1d_format.json
│   │   │   └── indices_format.json
│   │   └── system
│   │       ├── asset.json
│   │       ├── balance.json
│   │       ├── crypto_market.json
│   │       ├── currency_market.json
│   │       ├── indices_market.json
│   │       ├── market.json
│   │       ├── market_type.json
│   │       └── user.json
```

1. Project Description
ChatDB provides convenient information to users and manager from the database system used in robo-advisor named StradIAN

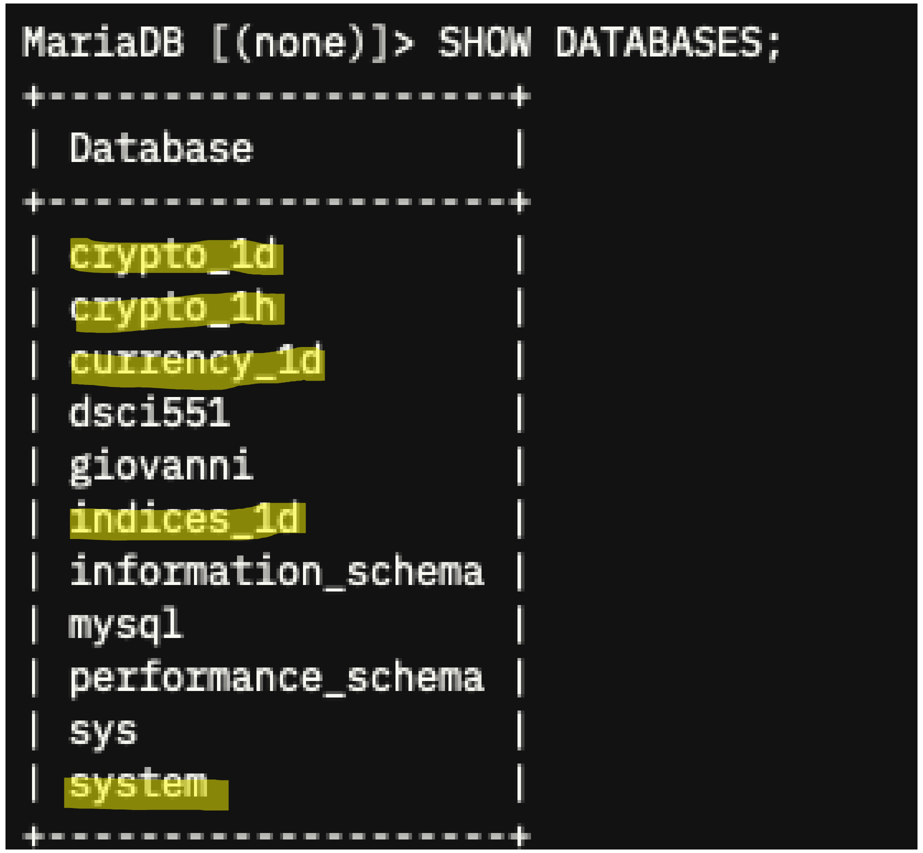
https://github.com/csian98/StradIAN

2. Development environment
Linux x86_64
Mariadb from 11.6.2-MariaDB, client 15.2 for Linux (x86_64)
Python 3.12.7

About the Project

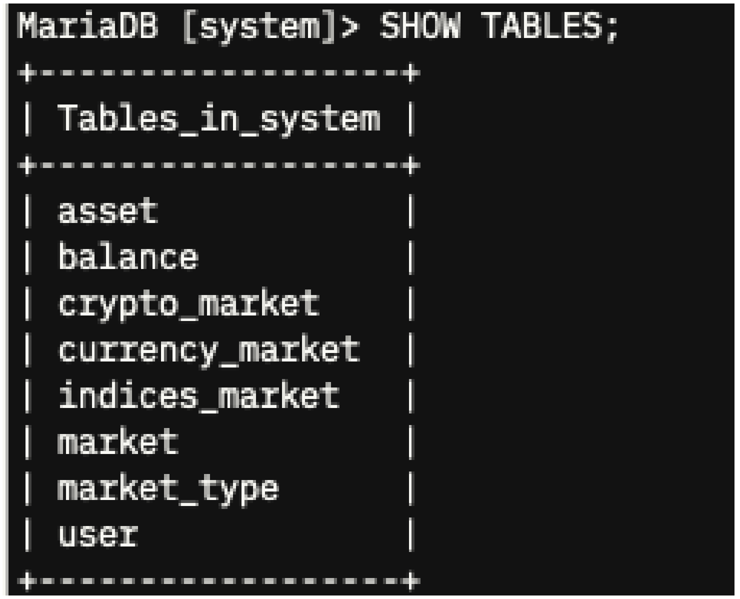
3. Dataset
- system
- crypto_1d
- crypto_1h
- currency_1d
- ~~indices_1d~~

4. Crawler
- pylib/stradian/crypto_web_crawler.py
(python3 pylib/exec/crypto_web_crawler.py)
- pylib/stradian/currency_web_crawler.py
(python3 pylib/exec/currency_web_crawler.py)
- pylib/stradian/indices_web_crawler.py
(python3 pylib/exec/indices_web_crawler.py)

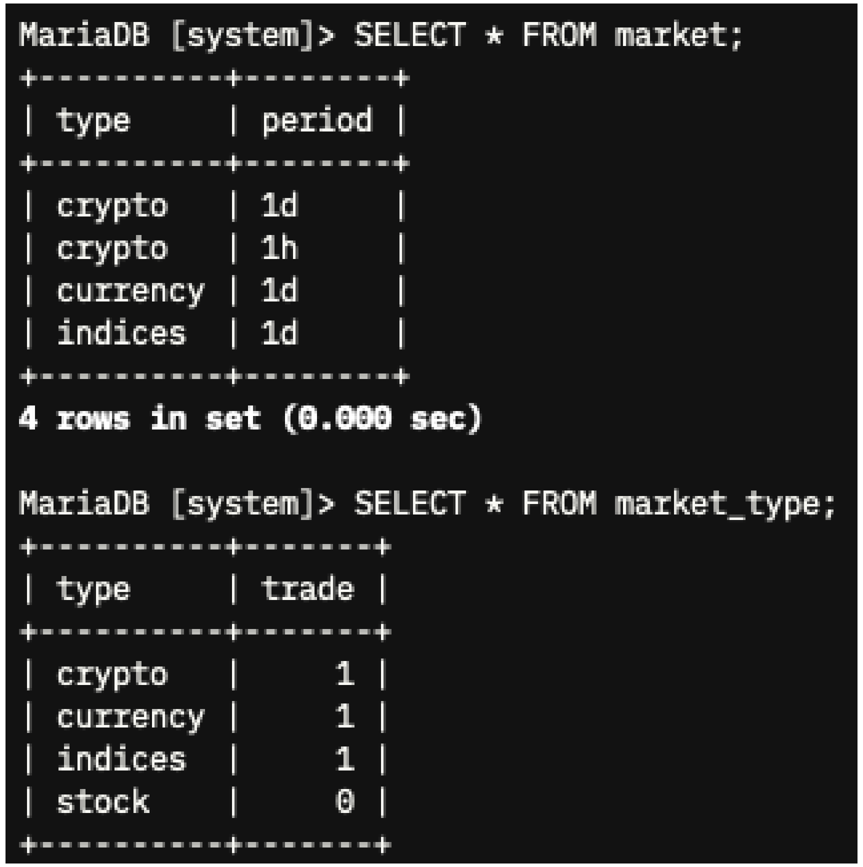


```
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| crypto_1d          |
| crypto_1h          |
| currency_1d        |
| dsci551            |
| giovanni           |
| indices_1d         |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| system             |
+--------------------+
```

About the Project

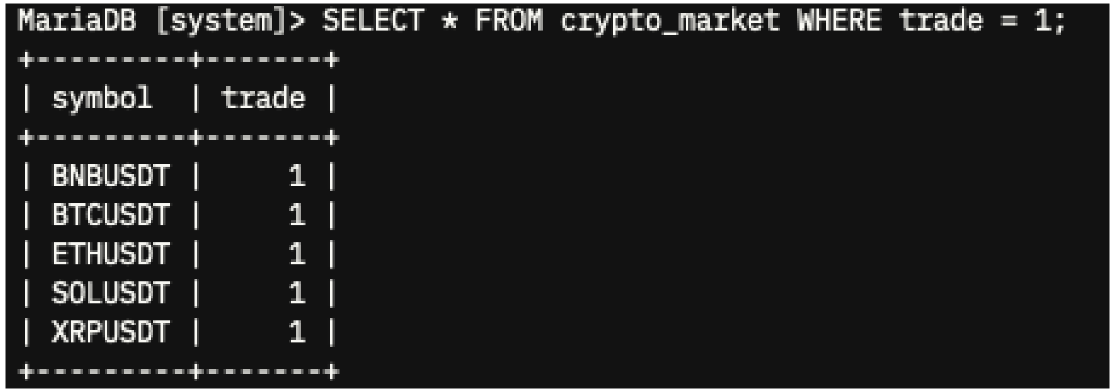


```
MariaDB [system]> SHOW TABLES;
+-------------------+
| Tables_in_system  |
+-------------------+
| asset             |
| balance           |
| crypto_market     |
| currency_market   |
| indices_market    |
| market            |
| market_type       |
| user              |
+-------------------+
```



```
MariaDB [system]> SELECT * FROM market;
+----------+--------+
| type     | period |
+----------+--------+
| crypto   | 1d     |
| crypto   | 1h     |
| currency | 1d     |
| indices  | 1d     |
+----------+--------+
4 rows in set (0.000 sec)

MariaDB [system]> SELECT * FROM market_type;
+----------+-------+
| type     | trade |
+----------+-------+
| crypto   |     1 |
| currency |     1 |
| indices  |     1 |
| stock    |     0 |
+----------+-------+
```

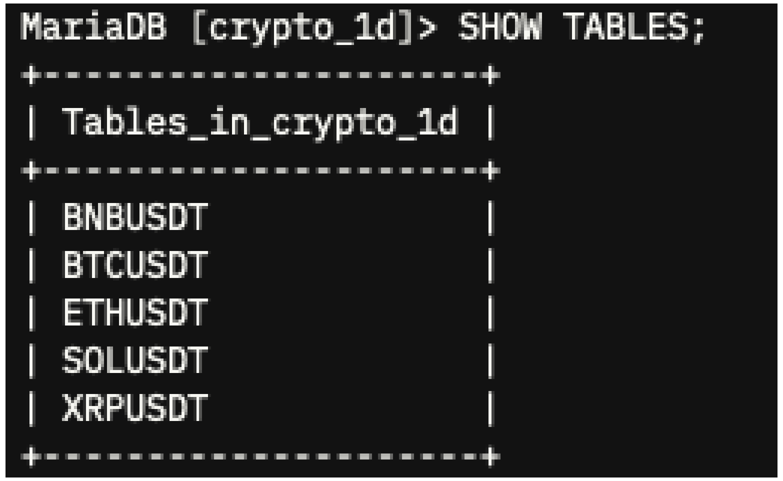
5. MariaDB Schema
Market information traded by StradIAN is stored in the **system.market** table,
And the database in use can be found through the **system.market_type** table.



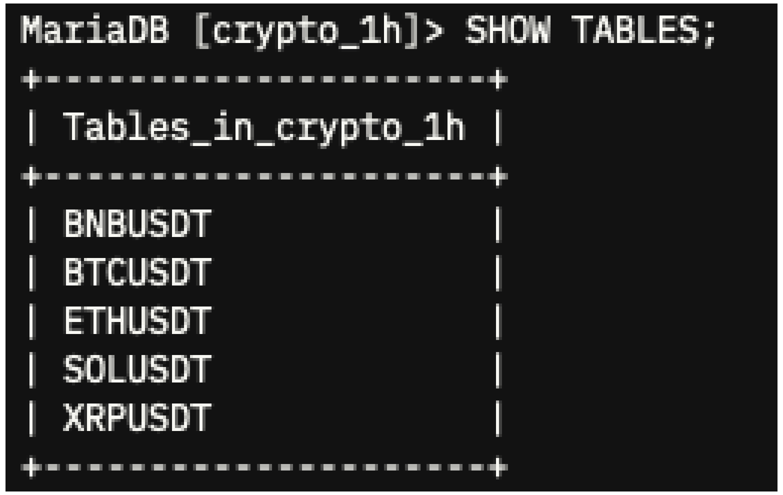
```
MariaDB [system]> SELECT * FROM crypto_market WHERE trade = 1;
+---------+-------+
| symbol  | trade |
+---------+-------+
| BNBUSDT |     1 |
| BTCUSDT |     1 |
| ETHUSDT |     1 |
| SOLUSDT |     1 |
| XRPUSDT |     1 |
+---------+-------+
```

The crawler stores data from the market, market_type,
and <market.type>_market tables as table in the
<market.type>_<market.period> database.

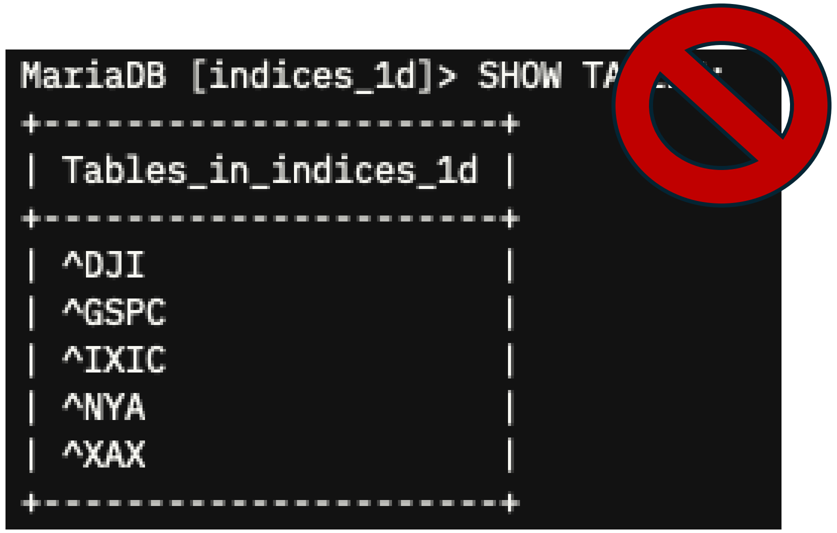
About the Project



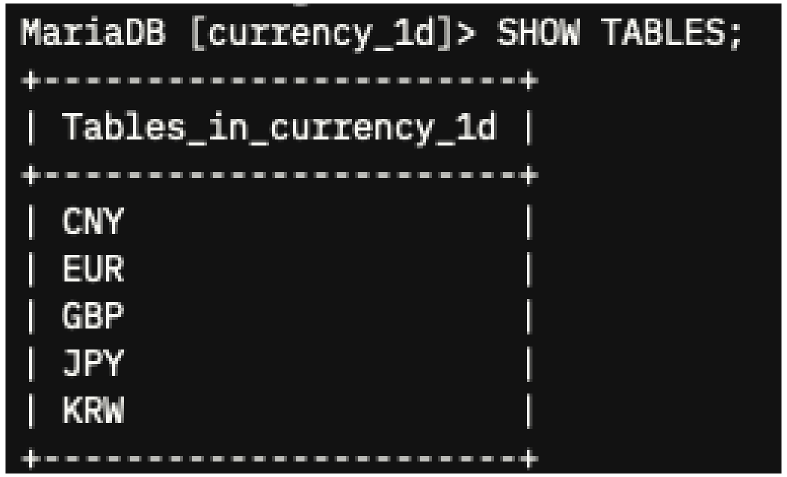
```
MariaDB [crypto_1d]> SHOW TABLES;
+---------------------+
| Tables_in_crypto_1d |
+---------------------+
| BNBUSDT             |
| BTCUSDT             |
| ETHUSDT             |
| SOLUSDT             |
| XRPUSDT             |
+---------------------+
```



```
MariaDB [crypto_1h]> SHOW TABLES;
+---------------------+
| Tables_in_crypto_1h |
+---------------------+
| BNBUSDT             |
| BTCUSDT             |
| ETHUSDT             |
| SOLUSDT             |
| XRPUSDT             |
+---------------------+
```



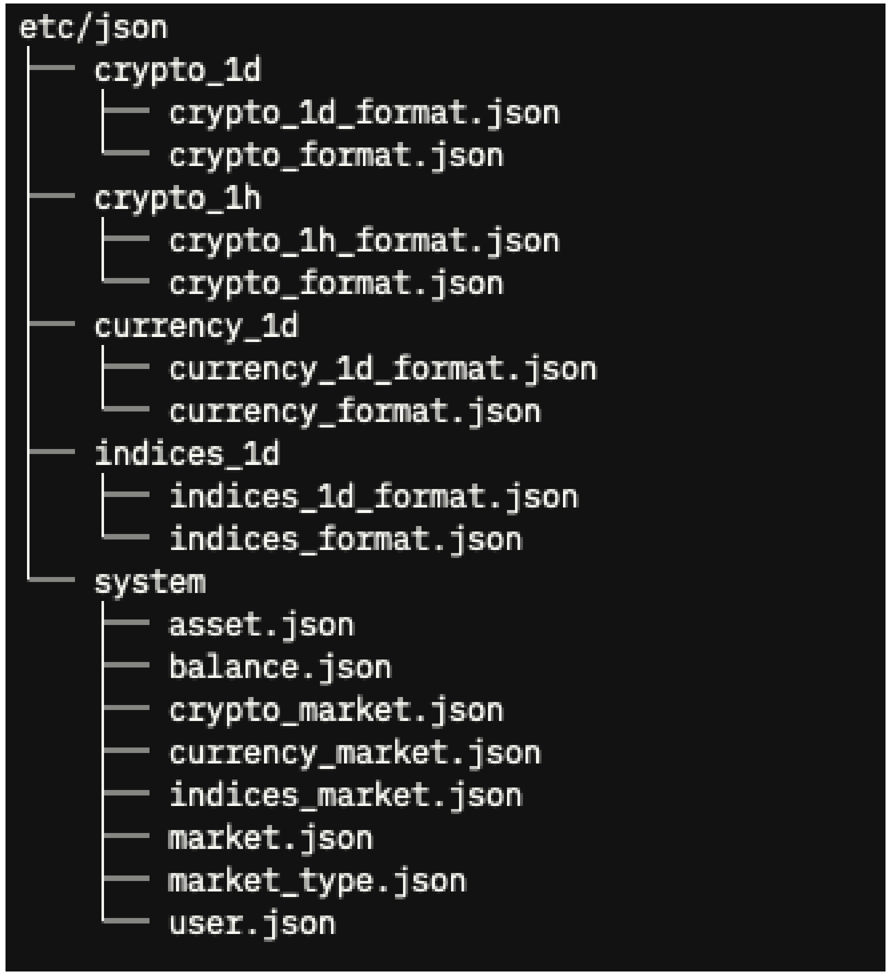
```
MariaDB [indices_1d]> SHOW TA
+----------------------+
| Tables_in_indices_1d |
+----------------------+
| ^DJI                 |
| ^GSPC                |
| ^IXIC                |
| ^NYA                 |
| ^XAX                 |
+----------------------+
```



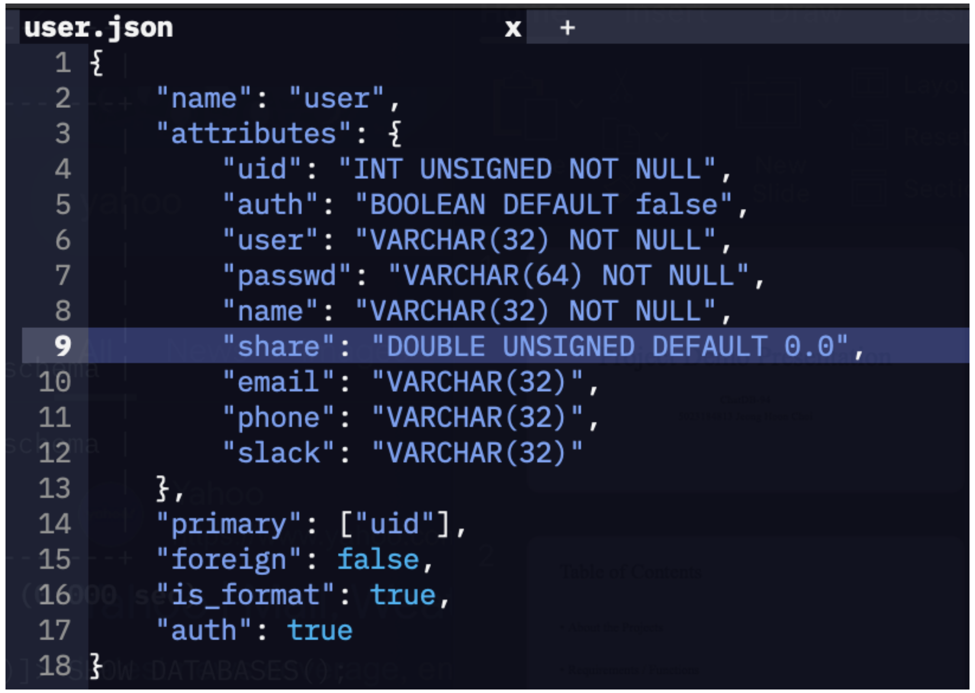
```
MariaDB [currency_1d]> SHOW TABLES;
+-----------------------+
| Tables_in_currency_1d |
+-----------------------+
| CNY                   |
| EUR                   |
| GBP                   |
| JPY                   |
| KRW                   |
+-----------------------+
```

Due to special characters in the table name of indices_1d table, the table name must be surrounded by backquotes(`). Not implemented yet.

About the Project



```
etc/json
├── crypto_1d
│   ├── crypto_1d_format.json
│   └── crypto_format.json
├── crypto_1h
│   ├── crypto_1h_format.json
│   └── crypto_format.json
├── currency_1d
│   ├── currency_1d_format.json
│   └── currency_format.json
├── indices_1d
│   ├── indices_1d_format.json
│   └── indices_format.json
└── system
    ├── asset.json
    ├── balance.json
    ├── crypto_market.json
    ├── currency_market.json
    ├── indices_market.json
    ├── market.json
    ├── market_type.json
    └── user.json
```



user.json

```
{
    "name": "user",
    "attributes": {
        "uid": "INT UNSIGNED NOT NULL",
        "auth": "BOOLEAN DEFAULT false",
        "user": "VARCHAR(32) NOT NULL",
        "passwd": "VARCHAR(64) NOT NULL",
        "name": "VARCHAR(32) NOT NULL",
        "share": "DOUBLE UNSIGNED DEFAULT 0.0",
        "email": "VARCHAR(32)",
        "phone": "VARCHAR(32)",
        "slack": "VARCHAR(32)"
    },
    "primary": ["uid"],
    "foreign": false,
    "is_format": true,
    "auth": true
}
```

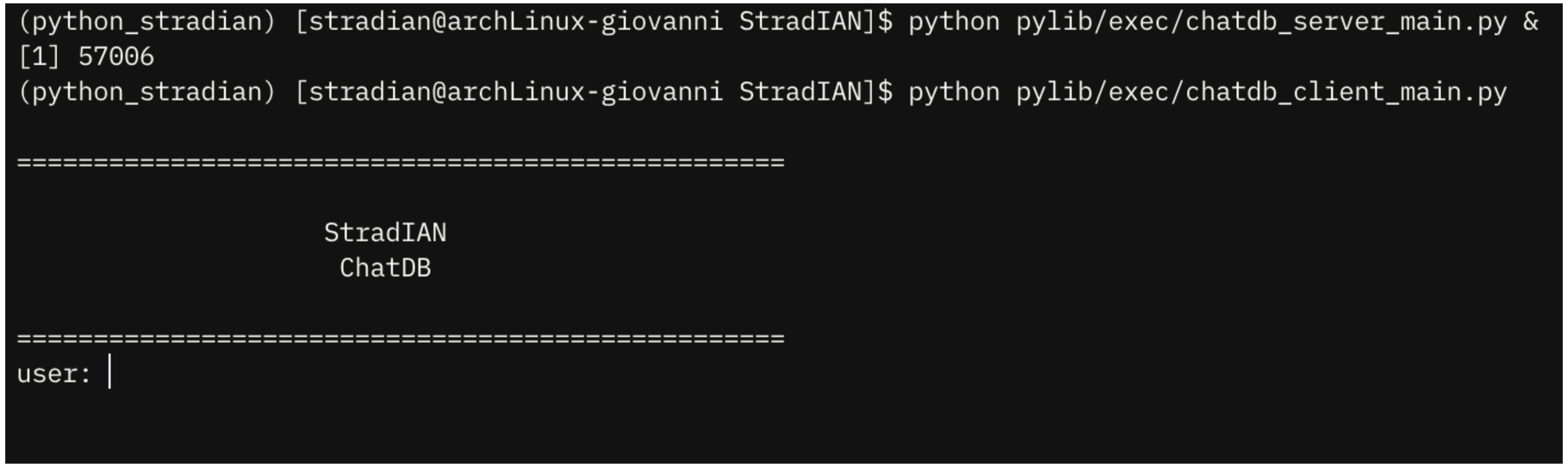
etc/json/system/user.json

Information about all table schemas is stored in **etc/json**

About the Project

Run server:
>>> python3 pylib/exec/chatdb_server_main.py &

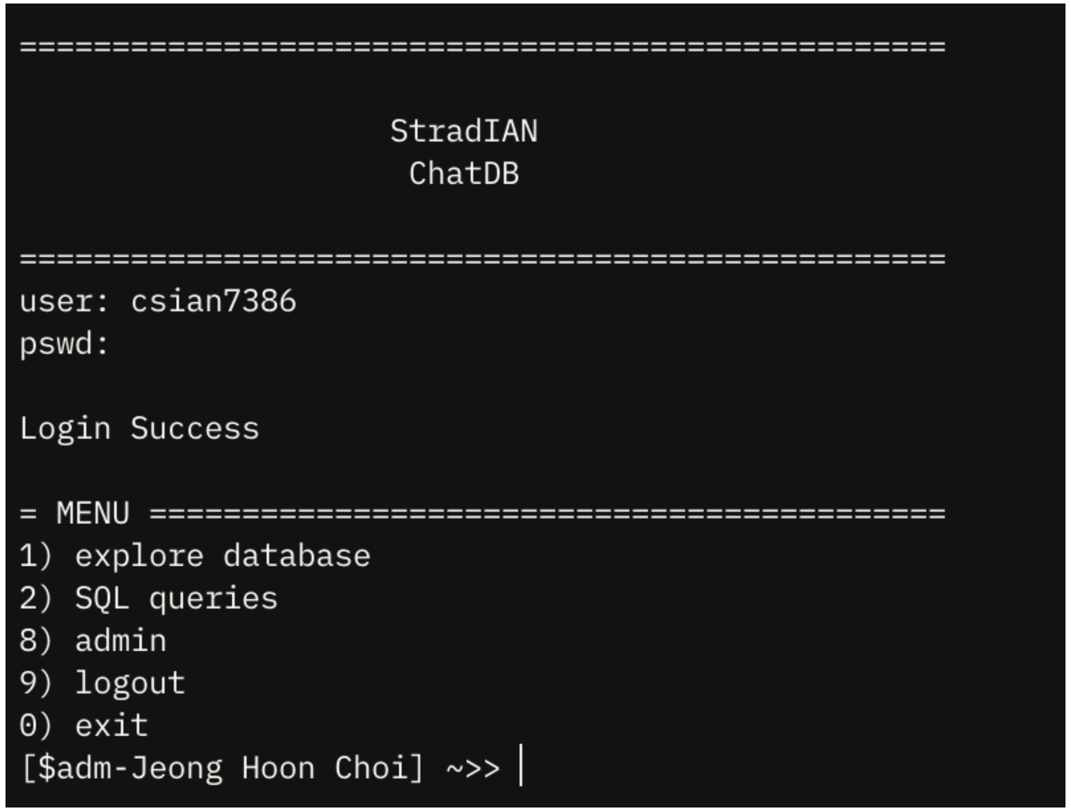
Run client:
>>> pyhton3 pylib/exec/chatdb_client_main.py



```
(python_stradian) [stradian@archLinux-giovanni StradIAN]$ python pylib/exec/chatdb_server_main.py &
[1] 57006
(python_stradian) [stradian@archLinux-giovanni StradIAN]$ python pylib/exec/chatdb_client_main.py

==================================================


                    StradIAN
                     ChatDB


==================================================
user: 
```

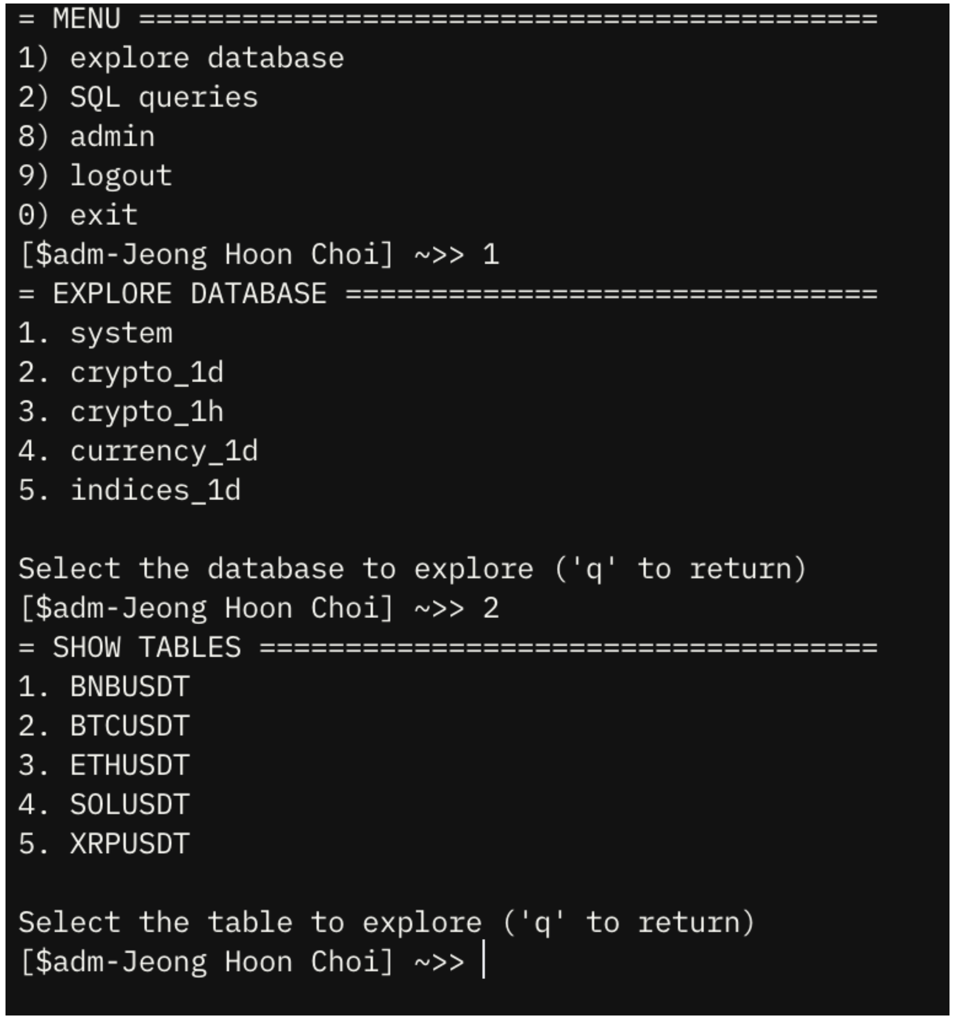
When logging in, different permissions are granted depending on the auth attribution in the system.user table.
(Hash the password and store it in the system.user table)

Explore Databases

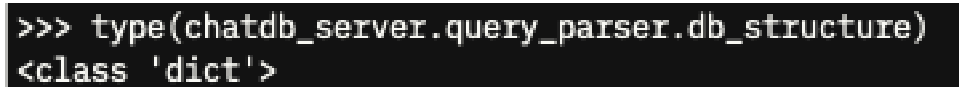


```
==================================================

                    StradIAN
                     ChatDB

==================================================
user: csian7386
pswd:

Login Success

= MENU ===========================================
1) explore database
2) SQL queries
8) admin
9) logout
0) exit
[$adm-Jeong Hoon Choi] ~>> 
```

Basically, the UI was designed using CLI.
(Implementing error handling for basic input errors)



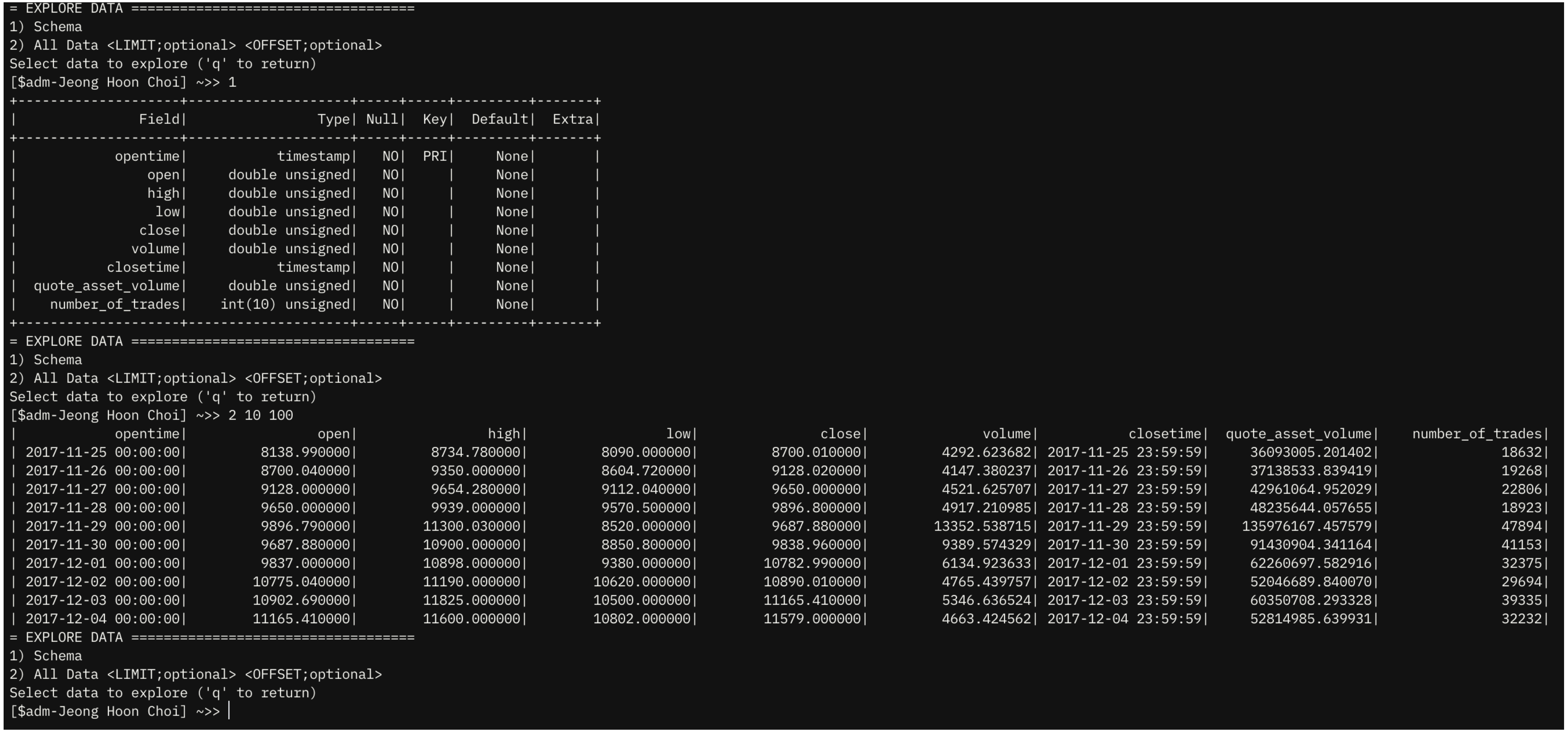
```
= MENU ===========================================
1) explore database
2) SQL queries
8) admin
9) logout
0) exit
[$adm-Jeong Hoon Choi] ~>> 1
= EXPLORE DATABASE ===============================
1. system
2. crypto_1d
3. crypto_1h
4. currency_1d
5. indices_1d

Select the database to explore ('q' to return)
[$adm-Jeong Hoon Choi] ~>> 2
= SHOW TABLES ====================================
1. BNBUSDT
2. BTCUSDT
3. ETHUSDT
4. SOLUSDT
5. XRPUSDT

Select the table to explore ('q' to return)
[$adm-Jeong Hoon Choi] ~>> 
```



```
>>> type(chatdb_server.query_parser.db_structure)
<class 'dict'>
```

Dictionary save
DB, Table, Attributes information

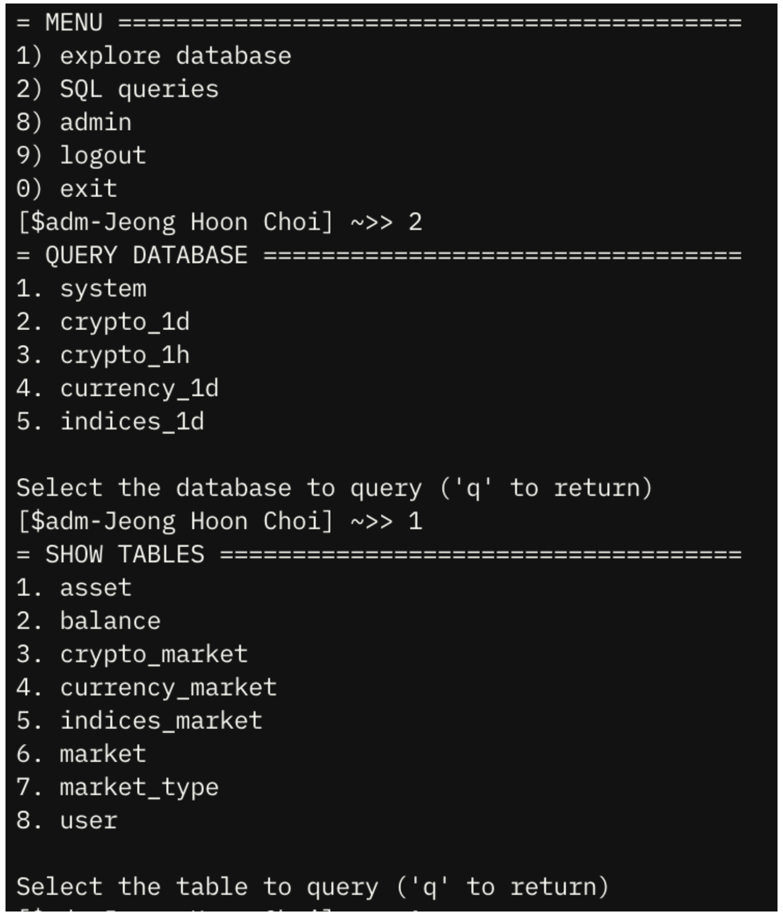
Explore Databases



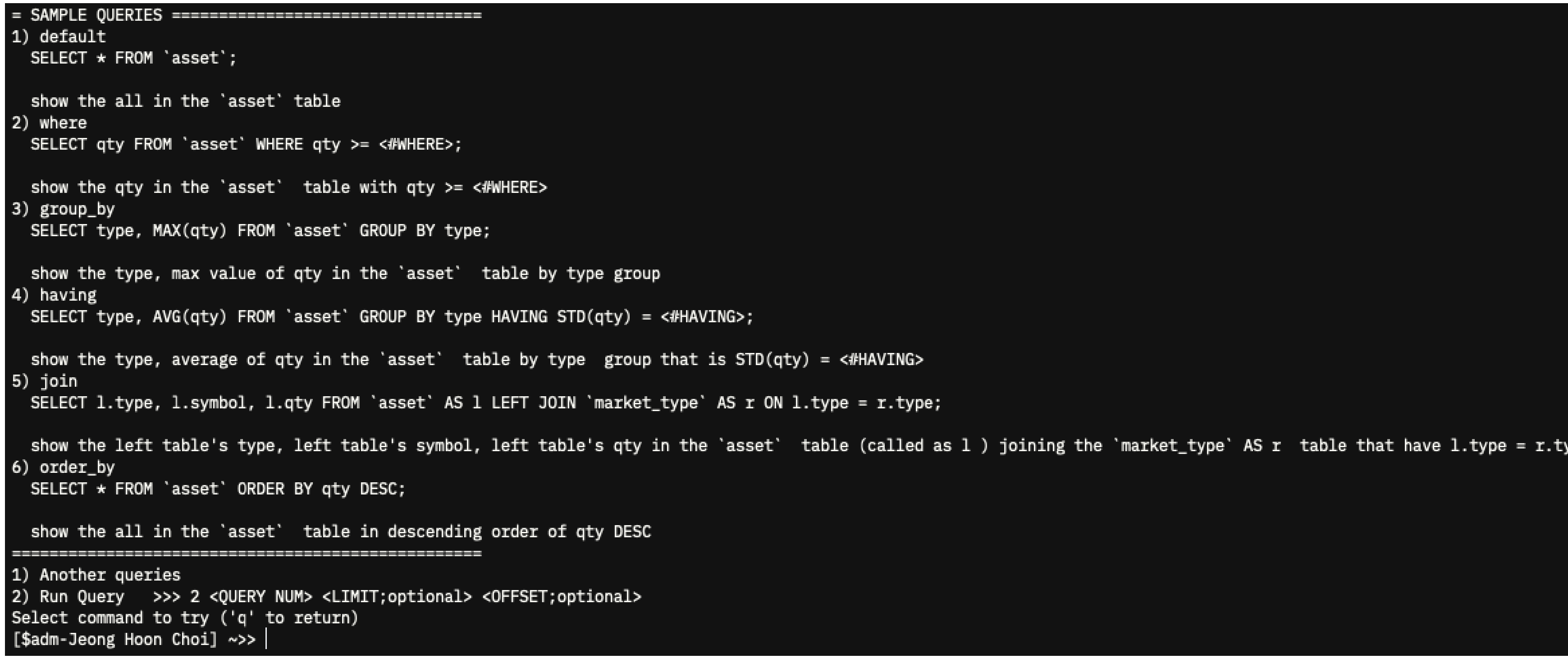
```
= EXPLORE DATA ===================================
1) Schema
2) All Data <LIMIT;optional> <OFFSET;optional>
Select data to explore ('q' to return)
[$adm-Jeong Hoon Choi] ~>> 1
+--------------------+--------------------+-----+-----+---------+-------+
|               Field|                Type| Null|  Key|  Default|  Extra|
+--------------------+--------------------+-----+-----+---------+-------+
|            opentime|           timestamp|   NO|  PRI|     None|       |
|                open|     double unsigned|   NO|     |     None|       |
|                high|     double unsigned|   NO|     |     None|       |
|                 low|     double unsigned|   NO|     |     None|       |
|               close|     double unsigned|   NO|     |     None|       |
|              volume|     double unsigned|   NO|     |     None|       |
|           closetime|           timestamp|   NO|     |     None|       |
|  quote_asset_volume|     double unsigned|   NO|     |     None|       |
|    number_of_trades|    int(10) unsigned|   NO|     |     None|       |
+--------------------+--------------------+-----+-----+---------+-------+
= EXPLORE DATA ===================================
1) Schema
2) All Data <LIMIT;optional> <OFFSET;optional>
Select data to explore ('q' to return)
[$adm-Jeong Hoon Choi] ~>> 2 10 100
|            opentime|                open|                high|                 low|               close|              volume|           closetime|  quote_asset_volume|    number_of_trades|
| 2017-11-25 00:00:00|         8138.990000|         8734.780000|         8090.000000|         8700.010000|         4292.623682| 2017-11-25 23:59:59|     36093005.201402|               18632|
| 2017-11-26 00:00:00|         8700.040000|         9350.000000|         8604.720000|         9128.020000|         4147.380237| 2017-11-26 23:59:59|     37138533.839419|               19268|
| 2017-11-27 00:00:00|         9128.000000|         9654.280000|         9112.040000|         9650.000000|         4521.625707| 2017-11-27 23:59:59|     42961064.952029|               22806|
| 2017-11-28 00:00:00|         9650.000000|         9939.000000|         9570.500000|         9896.800000|         4917.210985| 2017-11-28 23:59:59|     48235644.057655|               18923|
| 2017-11-29 00:00:00|         9896.790000|        11300.030000|         8520.000000|         9687.880000|        13352.538715| 2017-11-29 23:59:59|    135976167.457579|               47894|
| 2017-11-30 00:00:00|         9687.880000|        10900.000000|         8850.800000|         9838.960000|         9389.574329| 2017-11-30 23:59:59|     91430904.341164|               41153|
| 2017-12-01 00:00:00|         9837.000000|        10898.000000|         9380.000000|        10782.990000|         6134.923633| 2017-12-01 23:59:59|     62260697.582916|               32375|
| 2017-12-02 00:00:00|        10775.040000|        11190.000000|        10620.000000|        10890.010000|         4765.439757| 2017-12-02 23:59:59|     52046689.840070|               29694|
| 2017-12-03 00:00:00|        10902.690000|        11825.000000|        10500.000000|        11165.410000|         5346.636524| 2017-12-03 23:59:59|     60350708.293328|               39335|
| 2017-12-04 00:00:00|        11165.410000|        11600.000000|        10802.000000|        11579.000000|         4663.424562| 2017-12-04 23:59:59|     52814985.639931|               32232|
= EXPLORE DATA ===================================
1) Schema
2) All Data <LIMIT;optional> <OFFSET;optional>
Select data to explore ('q' to return)
[$adm-Jeong Hoon Choi] ~>> |
```

Schema search and data search for tables (limit, offset are optional) possible

Obtain Sample Queries



```
= MENU ==========================================
1) explore database
2) SQL queries
8) admin
9) logout
0) exit
[$adm-Jeong Hoon Choi] ~>> 2
= QUERY DATABASE ================================
1. system
2. crypto_1d
3. crypto_1h
4. currency_1d
5. indices_1d

Select the database to query ('q' to return)
[$adm-Jeong Hoon Choi] ~>> 1
= SHOW TABLES ===================================
1. asset
2. balance
3. crypto_market
4. currency_market
5. indices_market
6. market
7. market_type
8. user

Select the table to query ('q' to return)
```



```
= SAMPLE QUERIES ================================
1) default
  SELECT * FROM `asset`;

  show the all in the `asset` table
2) where
  SELECT qty FROM `asset` WHERE qty >= <#WHERE>;

  show the qty in the `asset`  table with qty >= <#WHERE>
3) group_by
  SELECT type, MAX(qty) FROM `asset` GROUP BY type;

  show the type, max value of qty in the `asset`  table by type group
4) having
  SELECT type, AVG(qty) FROM `asset` GROUP BY type HAVING STD(qty) = <#HAVING>;

  show the type, average of qty in the `asset`  table by type  group that is STD(qty) = <#HAVING>
5) join
  SELECT l.type, l.symbol, l.qty FROM `asset` AS l LEFT JOIN `market_type` AS r ON l.type = r.type;

  show the left table's type, left table's symbol, left table's qty in the `asset`  table (called as l ) joining the `market_type` AS r  table that have l.type = r.ty
6) order_by
  SELECT * FROM `asset` ORDER BY qty DESC;

  show the all in the `asset`  table in descending order of qty DESC
=================================================
1) Another queries
2) Run Query   >>> 2 <QUERY NUM> <LIMIT;optional> <OFFSET;optional>
Select command to try ('q' to return)
[$adm-Jeong Hoon Choi] ~>> |
```

pylib/stradian/query_parser.py class QueryParser

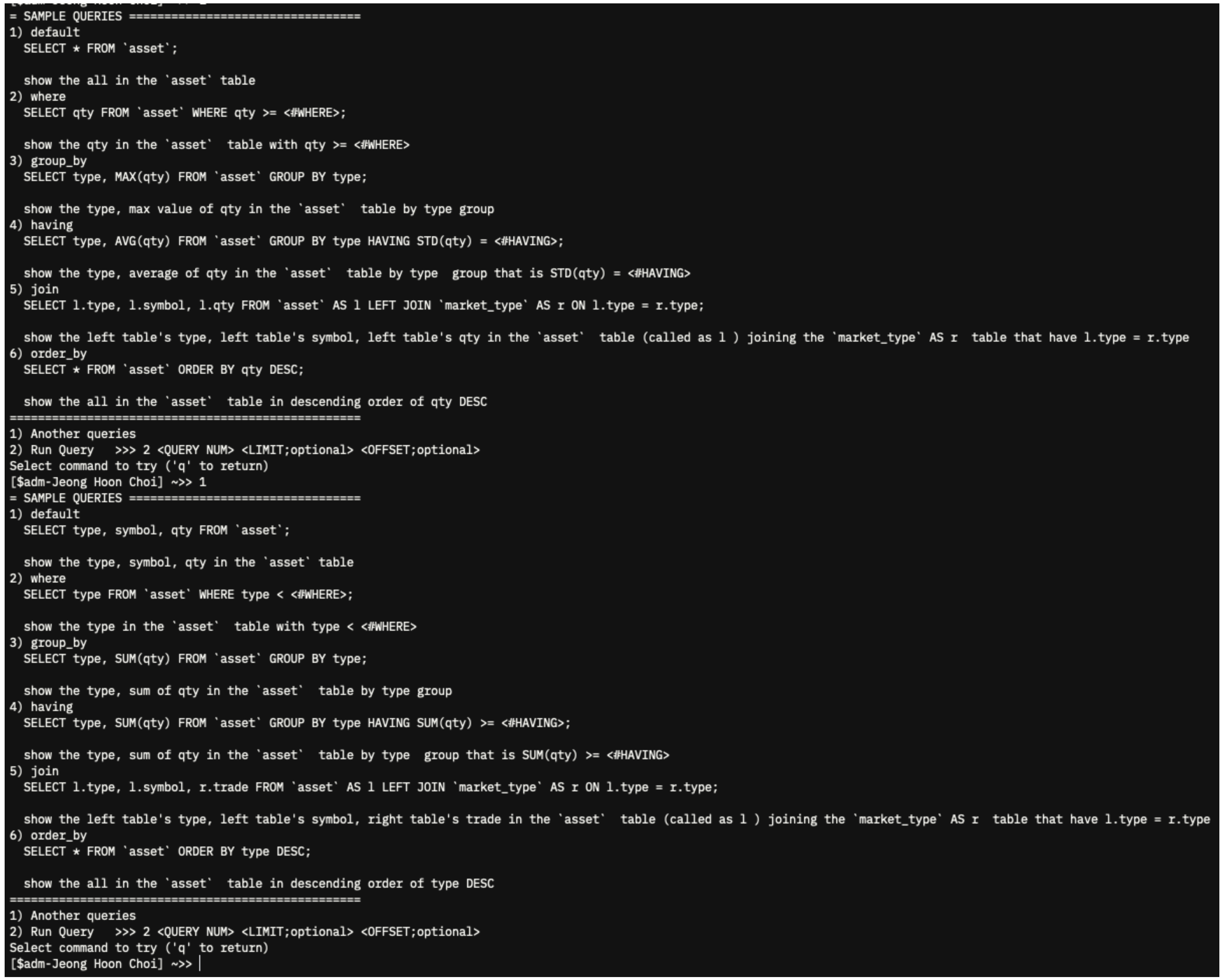
Random query statement and explaination output for 6 query_type

[“default”, “where”, “group_by”, “having”, “join”, “order_by”]

Referring to db_structure, if there are constraints such as group_by and join, the example queries of that type are not generated

Obtain Sample Queries

1) Another queries

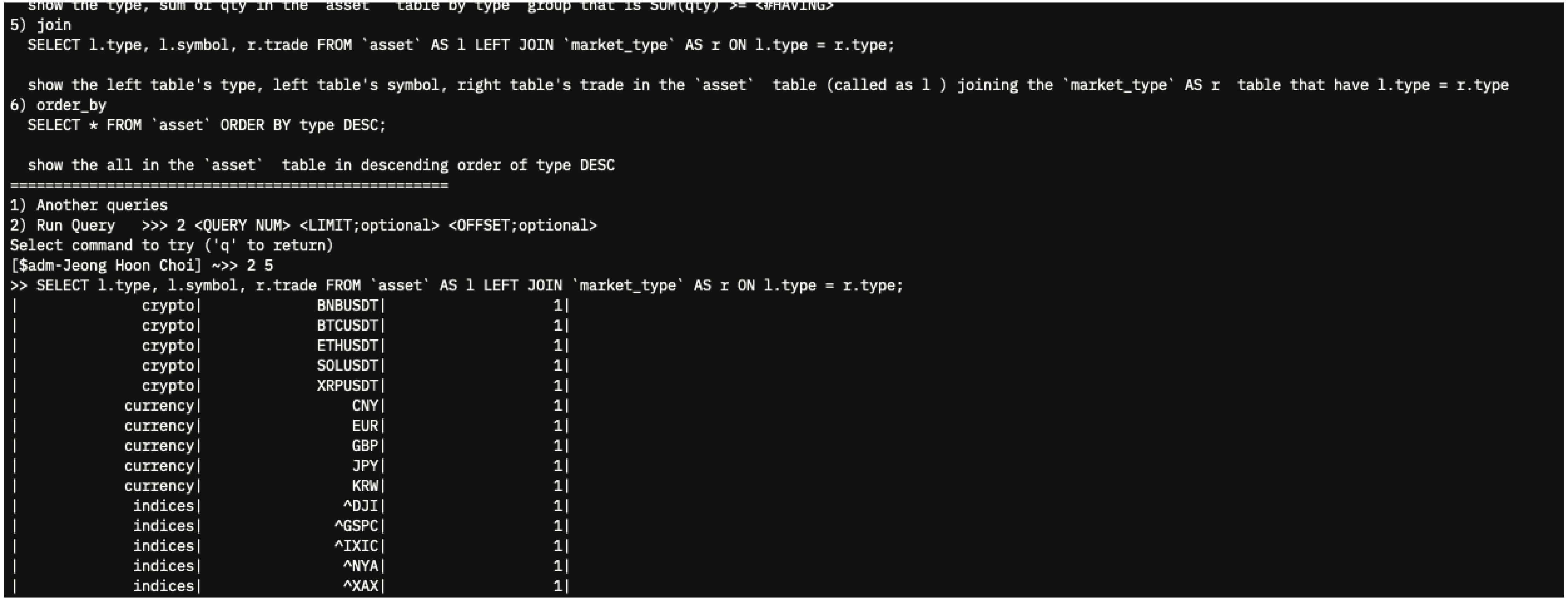


```
= SAMPLE QUERIES ================================
1) default
  SELECT * FROM `asset`;

  show the all in the `asset` table
2) where
  SELECT qty FROM `asset` WHERE qty >= <#WHERE>;

  show the qty in the `asset`  table with qty >= <#WHERE>
3) group_by
  SELECT type, MAX(qty) FROM `asset` GROUP BY type;

  show the type, max value of qty in the `asset`  table by type group
4) having
  SELECT type, AVG(qty) FROM `asset` GROUP BY type HAVING STD(qty) = <#HAVING>;

  show the type, average of qty in the `asset`  table by type  group that is STD(qty) = <#HAVING>
5) join
  SELECT l.type, l.symbol, l.qty FROM `asset` AS l LEFT JOIN `market_type` AS r ON l.type = r.type;

  show the left table's type, left table's symbol, left table's qty in the `asset`  table (called as l ) joining the `market_type` AS r  table that have l.type = r.type
6) order_by
  SELECT * FROM `asset` ORDER BY qty DESC;

  show the all in the `asset`  table in descending order of qty DESC
=================================================
1) Another queries
2) Run Query   >>> 2 <QUERY NUM> <LIMIT;optional> <OFFSET;optional>
Select command to try ('q' to return)
[$adm-Jeong Hoon Choi] ~>> 1
= SAMPLE QUERIES ================================
1) default
  SELECT type, symbol, qty FROM `asset`;

  show the type, symbol, qty in the `asset` table
2) where
  SELECT type FROM `asset` WHERE type < <#WHERE>;

  show the type in the `asset`  table with type < <#WHERE>
3) group_by
  SELECT type, SUM(qty) FROM `asset` GROUP BY type;

  show the type, sum of qty in the `asset`  table by type group
4) having
  SELECT type, SUM(qty) FROM `asset` GROUP BY type HAVING SUM(qty) >= <#HAVING>;

  show the type, sum of qty in the `asset`  table by type  group that is SUM(qty) >= <#HAVING>
5) join
  SELECT l.type, l.symbol, r.trade FROM `asset` AS l LEFT JOIN `market_type` AS r ON l.type = r.type;

  show the left table's type, left table's symbol, right table's trade in the `asset`  table (called as l ) joining the `market_type` AS r  table that have l.type = r.type
6) order_by
  SELECT * FROM `asset` ORDER BY type DESC;

  show the all in the `asset`  table in descending order of type DESC
=================================================
1) Another queries
2) Run Query   >>> 2 <QUERY NUM> <LIMIT;optional> <OFFSET;optional>
Select command to try ('q' to return)
[$adm-Jeong Hoon Choi] ~>> 
```

Obtain Sample Queries

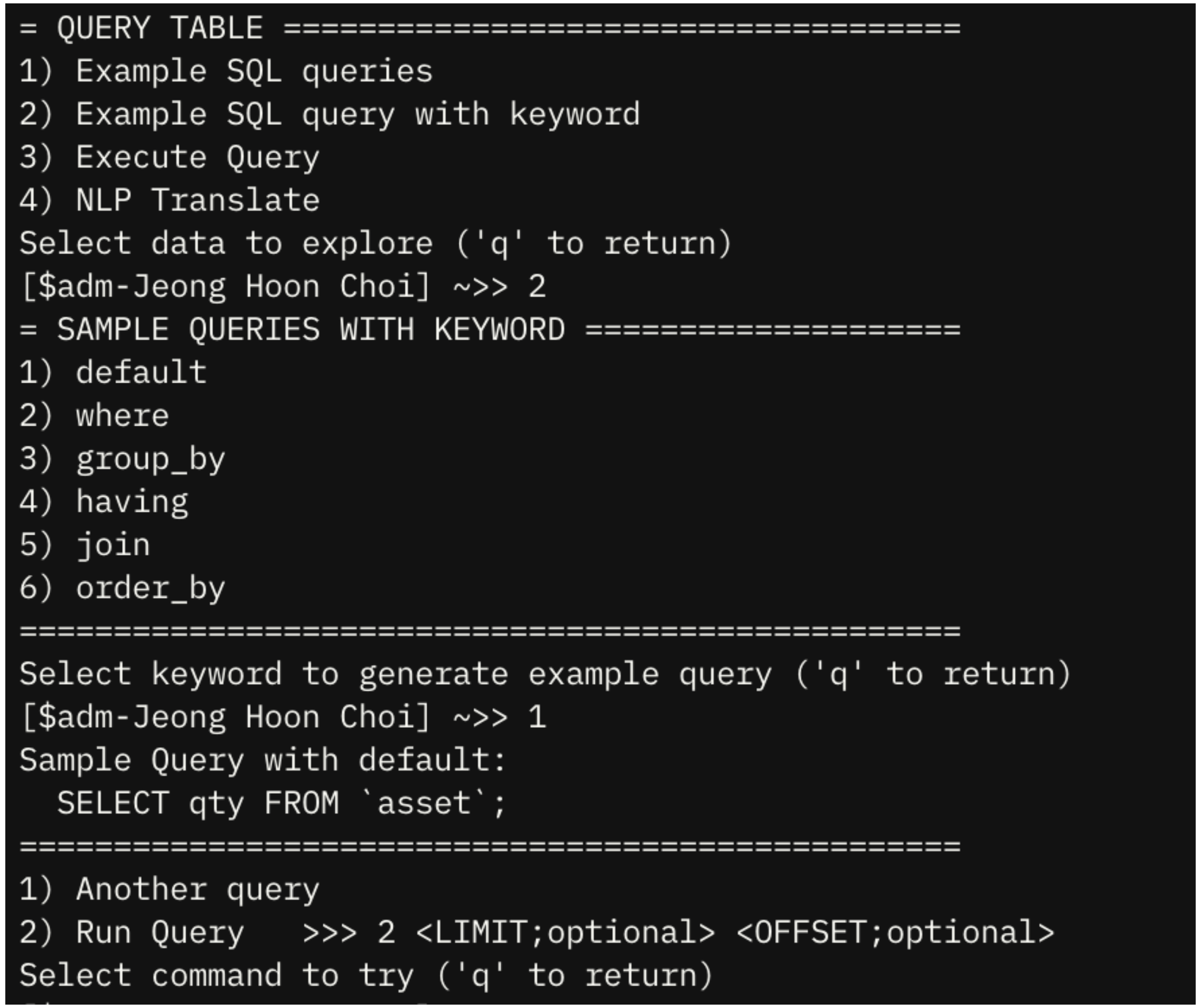
2) Run Query

<#WHERE>, <#HAVING> value can be selected or can be selected by random
(randomly choose from the database tables)

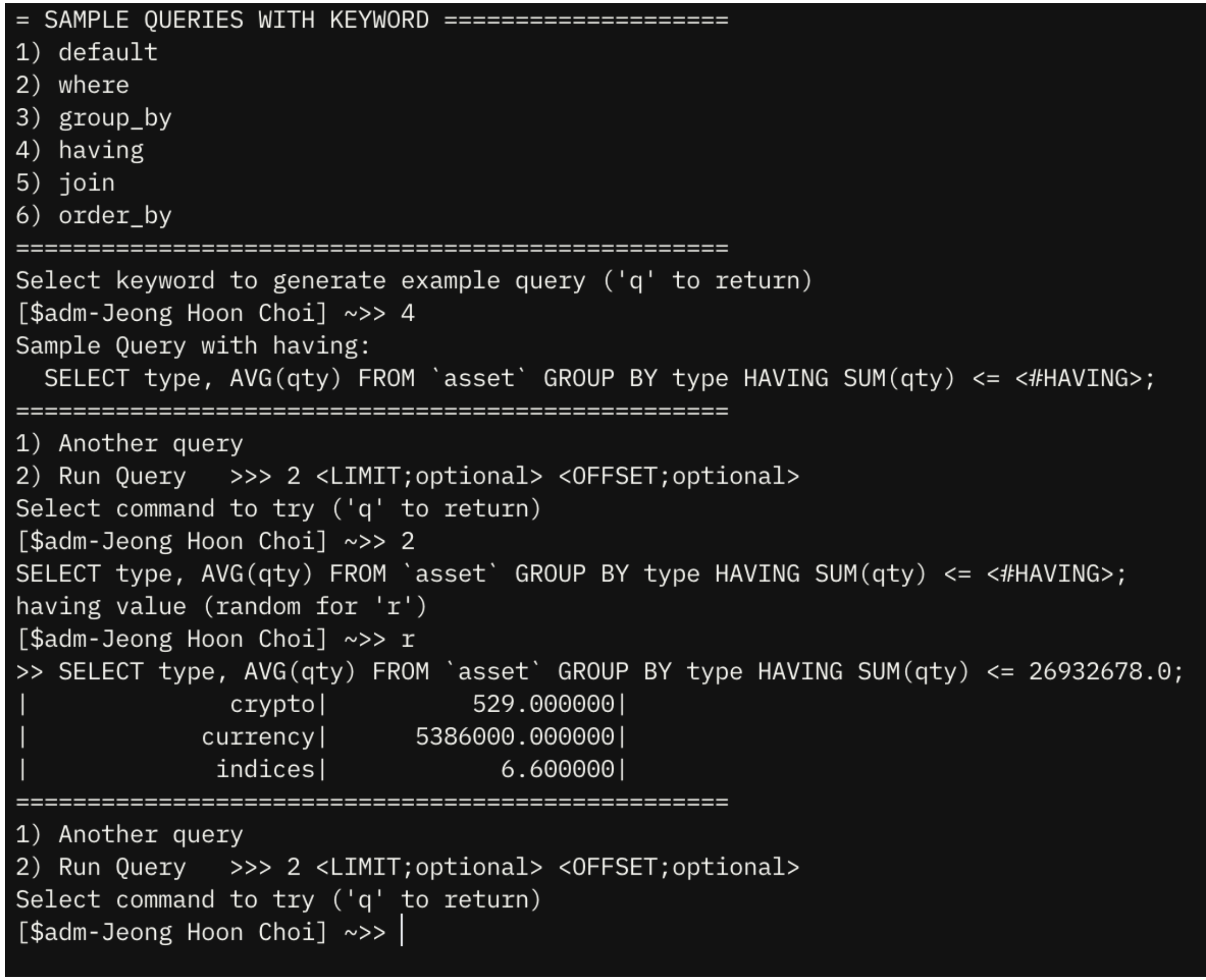


```
  show the type, sum of qty in the `asset`  table by type  group that is SUM(qty) >= <#HAVING>
5) join
  SELECT l.type, l.symbol, r.trade FROM `asset` AS l LEFT JOIN `market_type` AS r ON l.type = r.type;

  show the left table's type, left table's symbol, right table's trade in the `asset`  table (called as l ) joining the `market_type` AS r  table that have l.type = r.type
6) order_by
  SELECT * FROM `asset` ORDER BY type DESC;

  show the all in the `asset`  table in descending order of type DESC
==================================================
1) Another queries
2) Run Query   >>> 2 <QUERY NUM> <LIMIT;optional> <OFFSET;optional>
Select command to try ('q' to return)
[$adm-Jeong Hoon Choi] ~>> 2 5
>> SELECT l.type, l.symbol, r.trade FROM `asset` AS l LEFT JOIN `market_type` AS r ON l.type = r.type;
|              crypto|            BNBUSDT|                    1|
|              crypto|            BTCUSDT|                    1|
|              crypto|            ETHUSDT|                    1|
|              crypto|            SOLUSDT|                    1|
|              crypto|            XRPUSDT|                    1|
|            currency|                CNY|                    1|
|            currency|                EUR|                    1|
|            currency|                GBP|                    1|
|            currency|                JPY|                    1|
|            currency|                KRW|                    1|
|             indices|               ^DJI|                    1|
|             indices|              ^GSPC|                    1|
|             indices|              ^IXIC|                    1|
|             indices|               ^NYA|                    1|
|             indices|               ^XAX|                    1|
```

Obtain Sample Queries with Specific Language Constructs



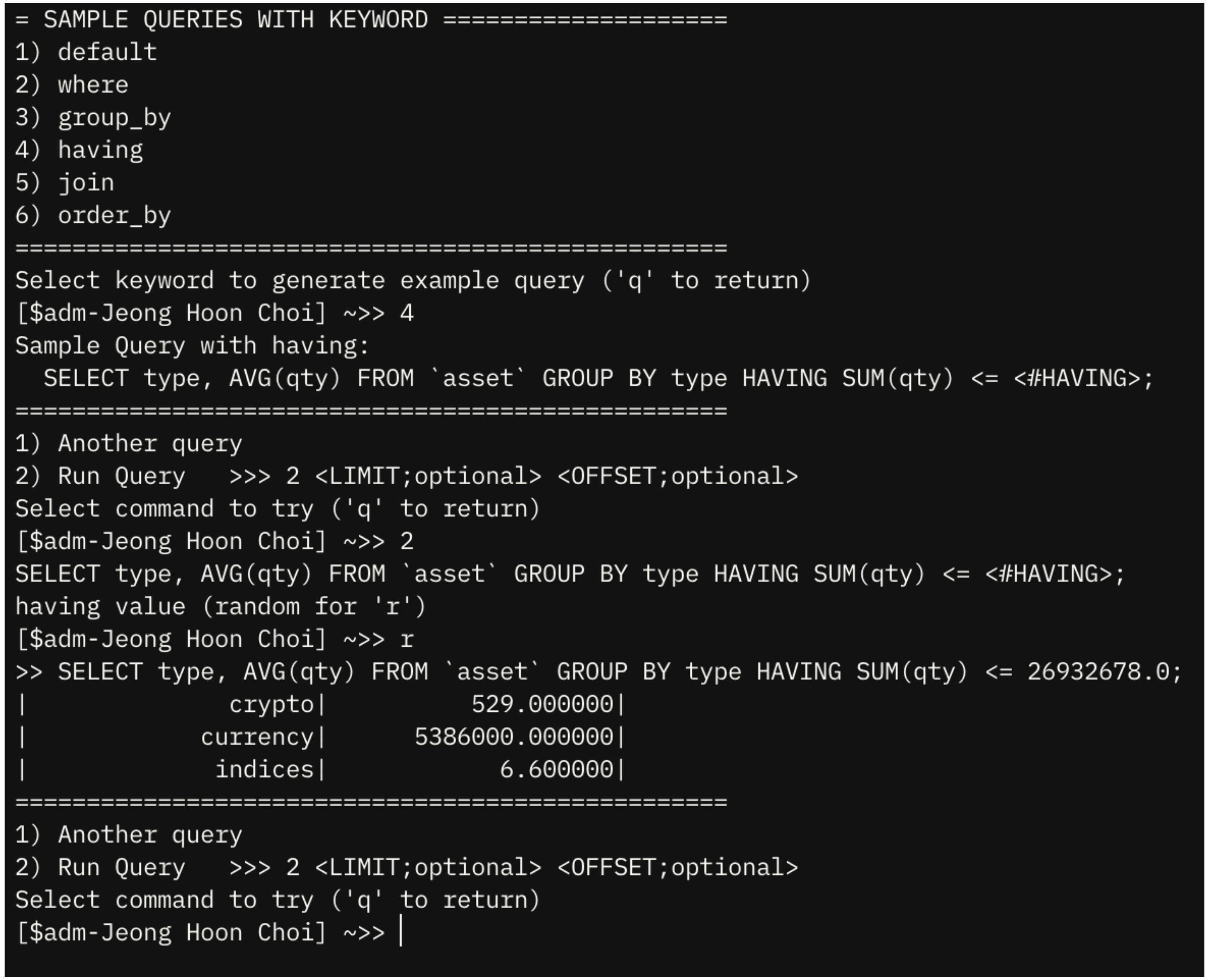
```
= QUERY TABLE ====================================
1) Example SQL queries
2) Example SQL query with keyword
3) Execute Query
4) NLP Translate
Select data to explore ('q' to return)
[$adm-Jeong Hoon Choi] ~>> 2
= SAMPLE QUERIES WITH KEYWORD ====================
1) default
2) where
3) group_by
4) having
5) join
6) order_by
==================================================
Select keyword to generate example query ('q' to return)
[$adm-Jeong Hoon Choi] ~>> 1
Sample Query with default:
  SELECT qty FROM `asset`;
==================================================
1) Another query
2) Run Query   >>> 2 <LIMIT;optional> <OFFSET;optional>
Select command to try ('q' to return)
```



```
= SAMPLE QUERIES WITH KEYWORD ====================
1) default
2) where
3) group_by
4) having
5) join
6) order_by
==================================================
Select keyword to generate example query ('q' to return)
[$adm-Jeong Hoon Choi] ~>> 4
Sample Query with having:
  SELECT type, AVG(qty) FROM `asset` GROUP BY type HAVING SUM(qty) <= <#HAVING>;
==================================================
1) Another query
2) Run Query   >>> 2 <LIMIT;optional> <OFFSET;optional>
Select command to try ('q' to return)
[$adm-Jeong Hoon Choi] ~>> 2
SELECT type, AVG(qty) FROM `asset` GROUP BY type HAVING SUM(qty) <= <#HAVING>;
having value (random for 'r')
[$adm-Jeong Hoon Choi] ~>> r
>> SELECT type, AVG(qty) FROM `asset` GROUP BY type HAVING SUM(qty) <= 26932678.0;
|              crypto|          529.000000|
|            currency|      5386000.000000|
|             indices|            6.600000|
==================================================
1) Another query
2) Run Query   >>> 2 <LIMIT;optional> <OFFSET;optional>
Select command to try ('q' to return)
[$adm-Jeong Hoon Choi] ~>>
```

Same with the “Sample query(Example SQL queries)” but specific keywords can be selected.

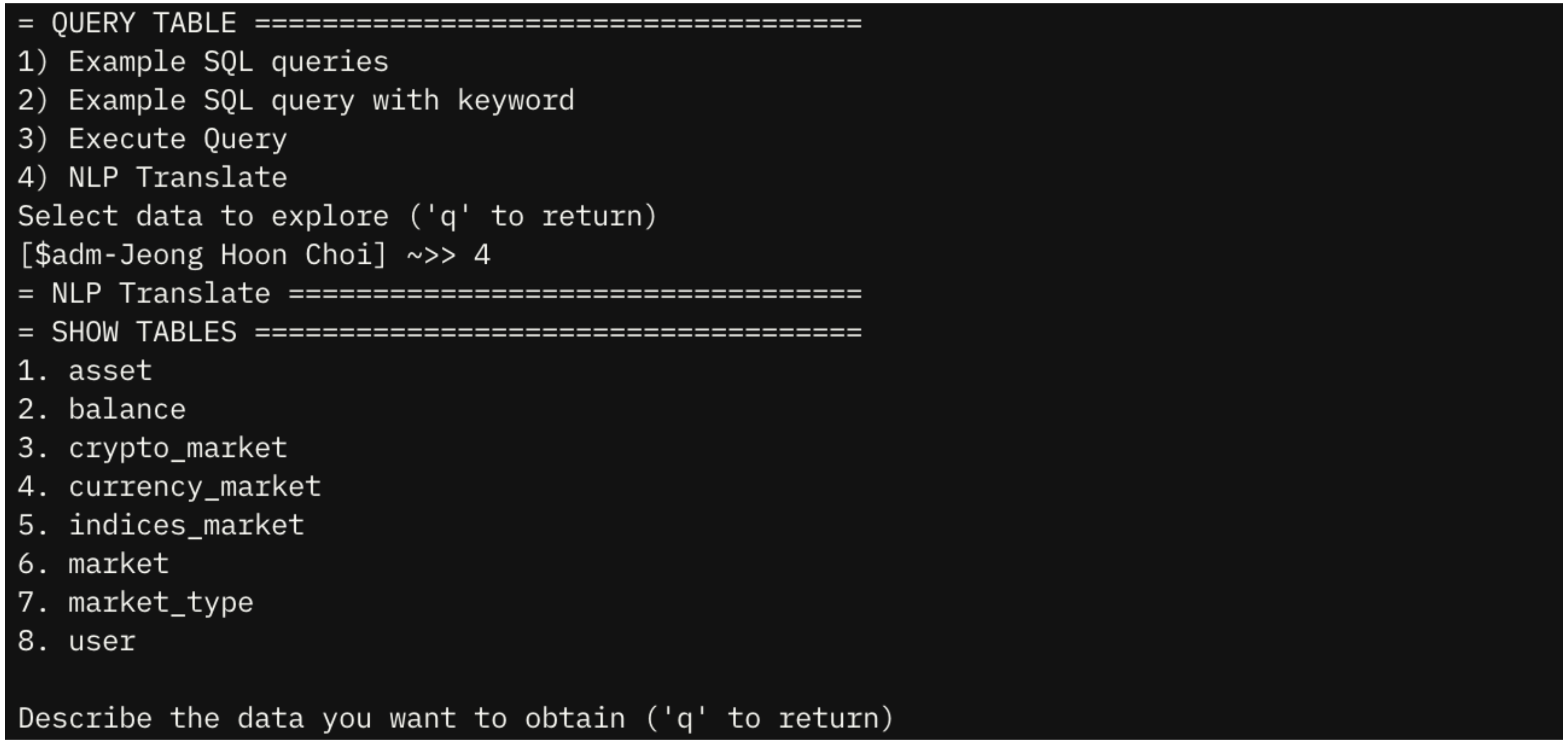
Obtain Sample Queries with Specific Language Constructs



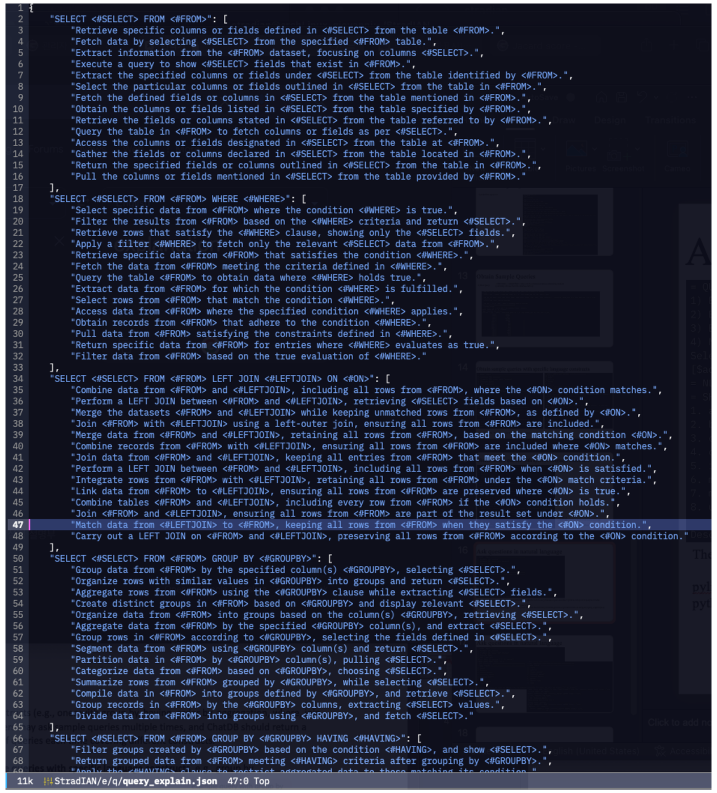
```
= SAMPLE QUERIES WITH KEYWORD ====================
1) default
2) where
3) group_by
4) having
5) join
6) order_by
==================================================
Select keyword to generate example query ('q' to return)
[$adm-Jeong Hoon Choi] ~>> 4
Sample Query with having:
  SELECT type, AVG(qty) FROM `asset` GROUP BY type HAVING SUM(qty) <= <#HAVING>;
==================================================
1) Another query
2) Run Query   >>> 2 <LIMIT;optional> <OFFSET;optional>
Select command to try ('q' to return)
[$adm-Jeong Hoon Choi] ~>> 2
SELECT type, AVG(qty) FROM `asset` GROUP BY type HAVING SUM(qty) <= <#HAVING>;
having value (random for 'r')
[$adm-Jeong Hoon Choi] ~>> r
>> SELECT type, AVG(qty) FROM `asset` GROUP BY type HAVING SUM(qty) <= 26932678.0;
|              crypto|          529.000000|
|            currency|      5386000.000000|
|             indices|            6.600000|
==================================================
1) Another query
2) Run Query   >>> 2 <LIMIT;optional> <OFFSET;optional>
Select command to try ('q' to return)
[$adm-Jeong Hoon Choi] ~>> |
```

Run Query
<#WHERE>, <#HAVING> value can be selected
or can be selected by random
(randomly choose from the database tables)

Ask questions in Natural Language



```
= QUERY TABLE ======================================
1) Example SQL queries
2) Example SQL query with keyword
3) Execute Query
4) NLP Translate
Select data to explore ('q' to return)
[$adm-Jeong Hoon Choi] ~>> 4
= NLP Translate ====================================
= SHOW TABLES ======================================
1. asset
2. balance
3. crypto_market
4. currency_market
5. indices_market
6. market
7. market_type
8. user

Describe the data you want to obtain ('q' to return)
```

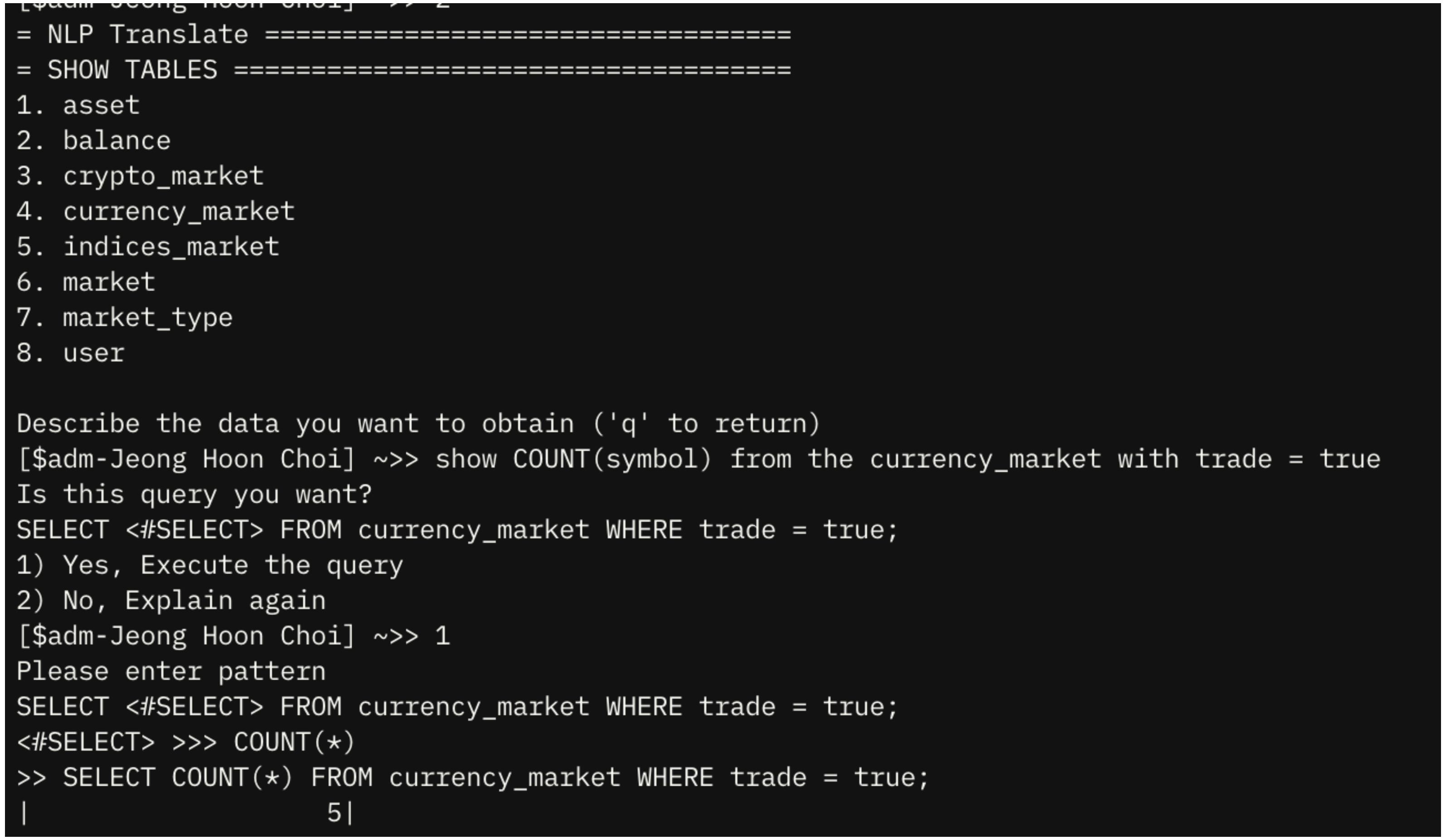


```
{
    "SELECT <#SELECT> FROM <#FROM>": [
        "Retrieve specific columns or fields defined in <#SELECT> from the table <#FROM>.",
        "Fetch data by selecting <#SELECT> from the specified <#FROM> table.",
        "Extract information from the <#FROM> dataset, focusing on columns <#SELECT>.",
        "Execute a query to show <#SELECT> fields that exist in <#FROM>.",
        "Extract the specified columns or fields under <#SELECT> from the table identified by <#FROM>.",
        "Select the particular columns or fields outlined in <#SELECT> from the table in <#FROM>.",
        "Fetch the defined fields or columns in <#SELECT> from the table mentioned in <#FROM>.",
        "Obtain the columns or fields listed in <#SELECT> from the table specified by <#FROM>.",
        "Retrieve the fields or columns stated in <#SELECT> from the table referred to by <#FROM>.",
        "Query the table in <#FROM> to fetch columns or fields as per <#SELECT>.",
        "Access the columns or fields designated in <#SELECT> from the table at <#FROM>.",
        "Gather the fields or columns declared in <#SELECT> from the table located in <#FROM>.",
        "Return the specified fields or columns outlined in <#SELECT> from the table in <#FROM>.",
        "Pull the columns or fields mentioned in <#SELECT> from the table provided by <#FROM>."
    ],
    "SELECT <#SELECT> FROM <#FROM> WHERE <#WHERE>": [
        "Select specific data from <#FROM> where the condition <#WHERE> is true.",
        "Filter the results from <#FROM> based on the <#WHERE> criteria and return <#SELECT>.",
        "Retrieve rows that satisfy the <#WHERE> clause, showing only the <#SELECT> fields.",
        "Apply a filter <#WHERE> to fetch only the relevant <#SELECT> data from <#FROM>.",
        "Retrieve specific data from <#FROM> that satisfies the condition <#WHERE>.",
        "Fetch the data from <#FROM> meeting the criteria defined in <#WHERE>.",
        "Query the table <#FROM> to obtain data where <#WHERE> holds true.",
        "Extract data from <#FROM> for which the condition <#WHERE> is fulfilled.",
        "Select rows from <#FROM> that match the condition <#WHERE>.",
        "Access data from <#FROM> where the specified condition <#WHERE> applies.",
        "Obtain records from <#FROM> that adhere to the condition <#WHERE>.",
        "Pull data from <#FROM> satisfying the constraints defined in <#WHERE>.",
        "Return specific data from <#FROM> for entries where <#WHERE> evaluates as true.",
        "Filter data from <#FROM> based on the true evaluation of <#WHERE>."
    ],
    "SELECT <#SELECT> FROM <#FROM> LEFT JOIN <#LEFTJOIN> ON <#ON>": [
        "Combine data from <#FROM> and <#LEFTJOIN>, including all rows from <#FROM>, where the <#ON> condition matches.",
        "Perform a LEFT JOIN between <#FROM> and <#LEFTJOIN>, retrieving <#SELECT> fields based on <#ON>.",
        "Merge the datasets <#FROM> and <#LEFTJOIN> while keeping unmatched rows from <#FROM>, as defined by <#ON>.",
        "Join <#FROM> with <#LEFTJOIN> using a left-outer join, ensuring all rows from <#FROM> are included.",
        "Merge data from <#FROM> and <#LEFTJOIN>, retaining all rows from <#FROM>, based on the matching condition <#ON>.",
        "Combine records from <#FROM> with <#LEFTJOIN>, ensuring all rows from <#FROM> are included where <#ON> matches.",
        "Join data from <#FROM> and <#LEFTJOIN>, keeping all entries from <#FROM> that meet the <#ON> condition.",
        "Perform a LEFT JOIN between <#FROM> and <#LEFTJOIN>, including all rows from <#FROM> when <#ON> is satisfied.",
        "Integrate rows from <#FROM> with <#LEFTJOIN>, retaining all rows from <#FROM> under the <#ON> match criteria.",
        "Link data from <#FROM> to <#LEFTJOIN>, ensuring all rows from <#FROM> are preserved where <#ON> is true.",
        "Combine tables <#FROM> and <#LEFTJOIN>, including every row from <#FROM> if the <#ON> condition holds.",
        "Join <#FROM> and <#LEFTJOIN>, ensuring all rows from <#FROM> are part of the result set under <#ON>.",
        "Match data from <#LEFTJOIN> to <#FROM>, keeping all rows from <#FROM> when they satisfy the <#ON> condition.",
        "Carry out a LEFT JOIN on <#FROM> and <#LEFTJOIN>, preserving all rows from <#FROM> according to the <#ON> condition."
    ],
    "SELECT <#SELECT> FROM <#FROM> GROUP BY <#GROUPBY>": [
        "Group data from <#FROM> by the specified column(s) <#GROUPBY>, selecting <#SELECT>.",
        "Organize rows with similar values in <#GROUPBY> into groups and return <#SELECT>.",
        "Aggregate rows from <#FROM> using the <#GROUPBY> clause while extracting <#SELECT> fields.",
        "Create distinct groups in <#FROM> based on <#GROUPBY> and display relevant <#SELECT>.",
        "Organize data from <#FROM> into groups based on the column(s) <#GROUPBY>, retrieving <#SELECT>.",
        "Aggregate data from <#FROM> by the specified <#GROUPBY> column(s), and extract <#SELECT>.",
        "Group rows in <#FROM> according to <#GROUPBY>, selecting the fields defined in <#SELECT>.",
        "Segment data from <#FROM> using <#GROUPBY> column(s) and return <#SELECT>.",
        "Partition data in <#FROM> by <#GROUPBY> column(s), pulling <#SELECT>.",
        "Categorize data from <#FROM> based on <#GROUPBY>, choosing <#SELECT>.",
        "Summarize rows from <#FROM> grouped by <#GROUPBY>, while selecting <#SELECT>.",
        "Compile data in <#FROM> into groups defined by <#GROUPBY>, and retrieve <#SELECT>.",
        "Group records in <#FROM> by the <#GROUPBY> columns, extracting <#SELECT> values.",
        "Divide data from <#FROM> into groups using <#GROUPBY>, and fetch <#SELECT>."
    ],
    "SELECT <#SELECT> FROM <#FROM> GROUP BY <#GROUPBY> HAVING <#HAVING>": [
        "Filter groups created by <#GROUPBY> based on the condition <#HAVING>, and show <#SELECT>.",
        "Return grouped data from <#FROM> meeting <#HAVING> criteria after grouping by <#GROUPBY>.",
        "Apply the <#HAVING> clause to restrict aggregated data to those matching its condition."
 11k  StradIAN/e/q/query_explain.json  47:0 Top
```

The query statement with the highest **Jaccard score** is output for a dictionary that uses a patterned description as a key and a query as a value.

pylib/stradian/query_parser.py class QueryParser **query_hash**
python3 pylib/exec/random_hash.py # parse and store in dictionary (random generate explanation and use etc/query/query_explain.json)

Ask questions in Natural Language

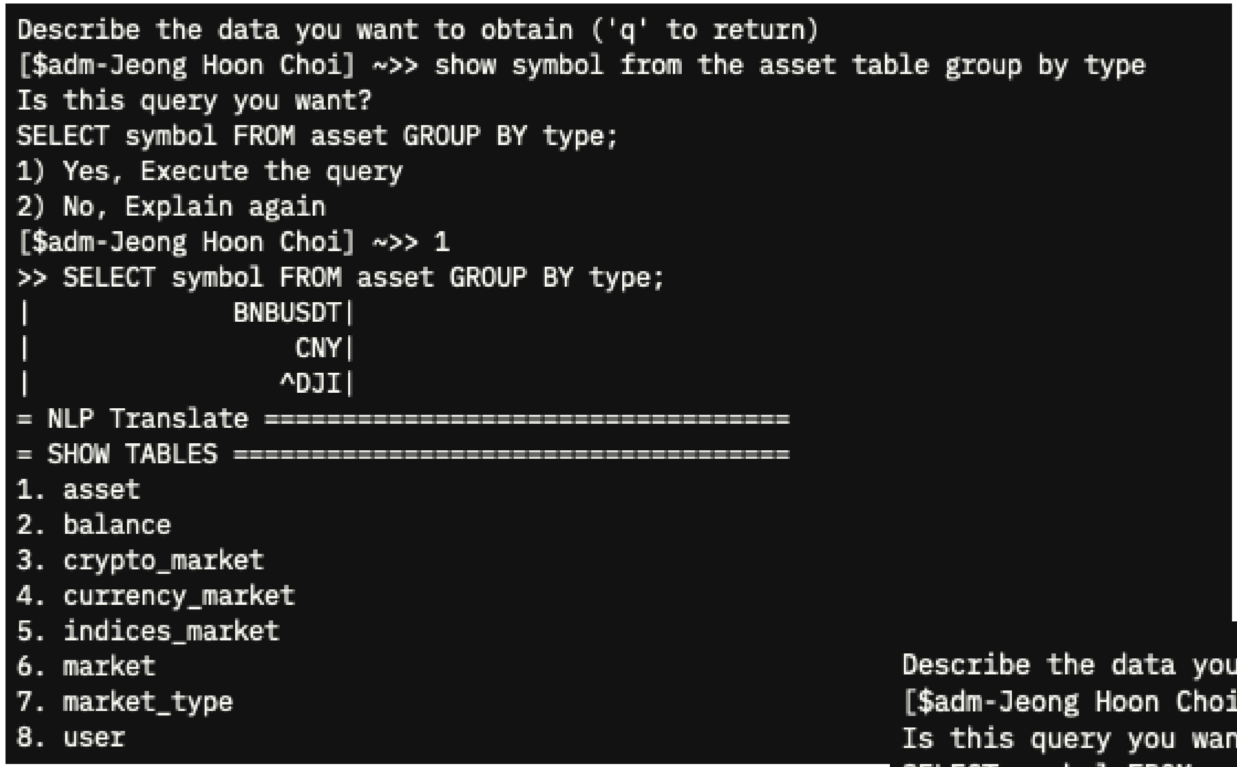


```
[$adm-Jeong Hoon Choi] ~>> 2
= NLP Translate ==================================
= SHOW TABLES ====================================
1. asset
2. balance
3. crypto_market
4. currency_market
5. indices_market
6. market
7. market_type
8. user

Describe the data you want to obtain ('q' to return)
[$adm-Jeong Hoon Choi] ~>> show COUNT(symbol) from the currency_market with trade = true
Is this query you want?
SELECT <#SELECT> FROM currency_market WHERE trade = true;
1) Yes, Execute the query
2) No, Explain again
[$adm-Jeong Hoon Choi] ~>> 1
Please enter pattern
SELECT <#SELECT> FROM currency_market WHERE trade = true;
<#SELECT> >>> COUNT(*)
>> SELECT COUNT(*) FROM currency_market WHERE trade = true;
|                   5|
```

SELECT <#SELECT> FROM <#FROM> AS <#AS> LEFT JOIN <#LEFTJOIN> AS <#AS> ON <#ON> WHERE <#WHERE> GROUP BY <#GROUPBY> HAVING <#HAVING> ORDER BY <#ORDERBY> (DESC) LIMIT <#LIMIT> OFFSET <#OFFSET>

Infer as much as possible about the <#*> pattern from the Natural Language description and return it.

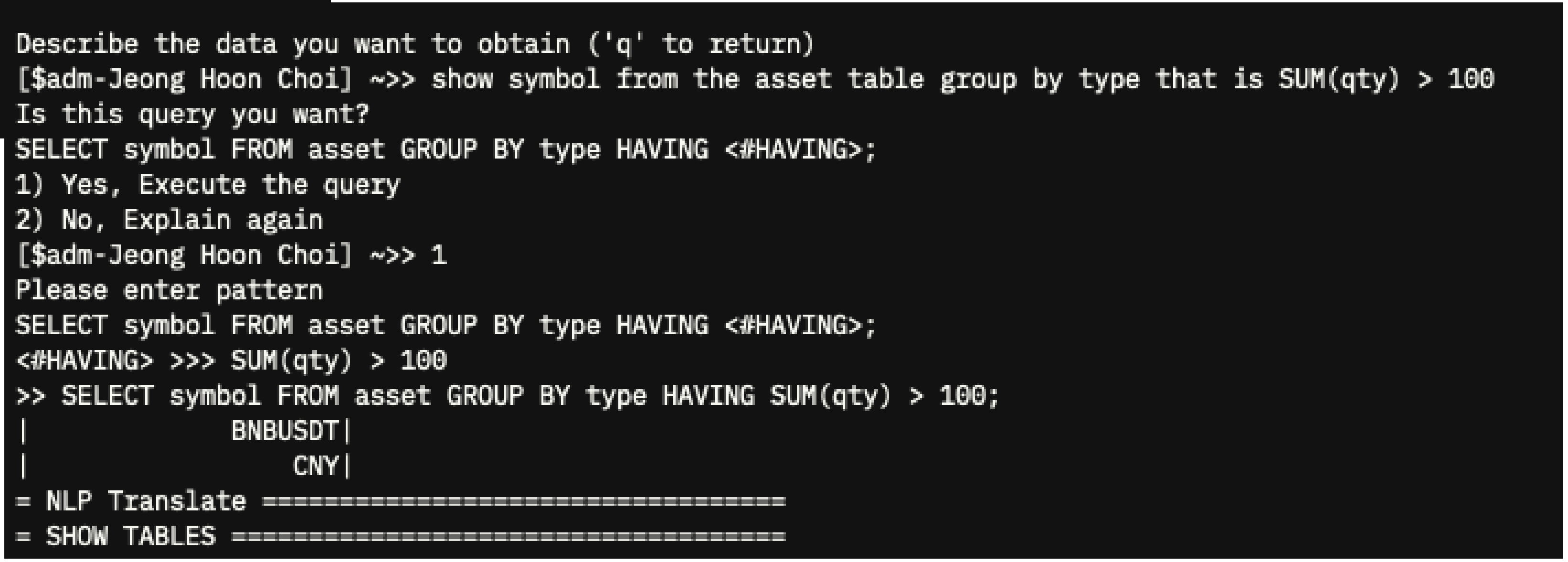
Ask questions in Natural Language



```
Describe the data you want to obtain ('q' to return)
[$adm-Jeong Hoon Choi] ~>> show symbol from the asset table group by type
Is this query you want?
SELECT symbol FROM asset GROUP BY type;
1) Yes, Execute the query
2) No, Explain again
[$adm-Jeong Hoon Choi] ~>> 1
>> SELECT symbol FROM asset GROUP BY type;
|              BNBUSDT|
|                  CNY|
|                 ^DJI|
= NLP Translate ===================================
= SHOW TABLES =====================================
1. asset
2. balance
3. crypto_market
4. currency_market
5. indices_market
6. market
7. market_type
8. user
```

Group By

Having



```
Describe the data you want to obtain ('q' to return)
[$adm-Jeong Hoon Choi] ~>> show symbol from the asset table group by type that is SUM(qty) > 100
Is this query you want?
SELECT symbol FROM asset GROUP BY type HAVING <#HAVING>;
1) Yes, Execute the query
2) No, Explain again
[$adm-Jeong Hoon Choi] ~>> 1
Please enter pattern
SELECT symbol FROM asset GROUP BY type HAVING <#HAVING>;
<#HAVING> >>> SUM(qty) > 100
>> SELECT symbol FROM asset GROUP BY type HAVING SUM(qty) > 100;
|             BNBUSDT|
|                 CNY|
= NLP Translate ===================================
= SHOW TABLES =====================================
```

Thank you