Natural Language to SQL Query System for Financial Data

Tianyu Shi, Boyang Lyu
Section: 615
jamiecoolby001@gmail.com
shitianyuxxgh@gmail.com

- **GitHub Repository:** https://github.com/Tianyu888/dbproject
- **Description:** This project focuses on converting natural language queries into SQL statements for financial data analysis, enabling seamless interactions with financial databases.

Project Background

In the current data-driven financial landscape, accessing and analyzing high-quality financial data is a crucial requirement for investors and researchers. However, for users lacking SQL knowledge or database operation skills, quickly accessing this data remains a significant challenge. Existing mainstream financial data platforms, such as Bloomberg Terminal and Yahoo Finance, provide professional-grade tools and data. However, these platforms are often complex, require specialized training, or impose high subscription fees, making them inaccessible or user-unfriendly for ordinary individuals.

To address this issue, we have built a financial database containing information on hundreds of companies and developed a solution leveraging Natural Language Processing (NLP) techniques to enable users to query this data using natural language.

Our database includes the following key tables:

- **Securities**: Stores information about securities, including codes, names, IPO dates, and exchange details.
- **Companies**: Contains company-level information, such as company names, registered capital, and employee counts.
- **Balance Sheet**: Covers key balance sheet data, such as total assets, current assets, and total liabilities.
- **Income Statement**: Includes company income statement data, such as operating revenue, operating costs, and net profit.
- **Financial Analysis**: Provides financial analysis metrics, including earnings per share, debt-to-equity ratio, and profit growth rate.
- **ESG**: Records environmental, social, and governance (ESG) ratings and related metrics for companies.

Our database is constructed using raw data sourced directly from Bloomberg. We preprocessed the raw data based on SQL standards to standardize its structure and optimize it for query performance.

Project Objectives

The primary objective of our project is to build a user-friendly platform that enables individuals with no SQL knowledge to effortlessly query financial data using natural language. To achieve this, we have employed the following architecture:

- **Django Framework**: Django serves as the backbone of our project, enabling seamless communication between the front-end and back-end. It provides a robust and secure web interface for users to input their natural language queries and view results.
- **OpenAI API Integration**: We utilize OpenAI's large language models (LLMs) through its API to handle natural language processing (NLP). These LLMs translate user queries into accurate SQL statements by understanding the context of the input and mapping it to the database schema.

- **MySQL Database**: Our preprocessed financial data, structured and standardized in SQL format, is hosted locally in a MySQL database. The translated SQL queries from the NLP model are executed on this database to retrieve the requested information.
- **User Interface**: The Django framework dynamically renders the results retrieved from the MySQL database and displays them on the user interface. This process ensures a smooth user experience, with no need for technical expertise.

We used a local MySQL server running on Python 3.11 with the `pymysql` library to connect the database to our Django application.

Workflow Overview

1. A user inputs a financial query (e.g., "Show me the top 10 companies by revenue last year").
2. The query is sent to the Django server, which communicates with OpenAI's API to translate it into SQL.
3. The SQL query is executed on the MySQL database to fetch relevant data.
4. The results are sent back to the Django front-end, where they are displayed in an intuitive, user-friendly format.

Areas of Specialization

Our project incorporates multiple specialized areas, including:

- **Natural Language Interfaces (6)**: Leveraging OpenAI's NLP models, we developed a system that dynamically converts natural language queries into SQL statements. This enables non-technical users to interact with complex financial data without needing expertise in database languages, significantly improving accessibility.
- **Datamining and Knowledge Discovery (7)**: By integrating a structured MySQL database of financial data, the project empowers users to uncover actionable insights from vast amounts of company-level information, such as revenue trends, ESG metrics, and industry comparisons.
- **User Interface Design (2)**: The Django-powered web application provides an intuitive and interactive user interface, streamlining the query process and visualizing results in a user-friendly format.
- **Data Modeling (3)**: The database schema was carefully designed to capture diverse financial metrics across multiple dimensions (e.g., balance sheets, income statements, and ESG ratings). This ensures the data is well-organized and optimized for fast, accurate querying.
- **Complex Data Extraction (1)**: Raw financial data was sourced from Bloomberg and preprocessed into SQL-compliant formats, ensuring compatibility and efficient storage for large-scale querying.

Project Strengths and Selling Points

- **Natural Language Accessibility**: Enables non-professional users to retrieve financial data effortlessly using plain language, removing the need for technical expertise.
- **Focus on Everyday Investors**: Designed for ordinary users who lack access to expensive tools, providing them with up-to-date financial insights for better decision-making.
- **Simplified Financial Data Analysis**: Streamlines the process of querying and visualizing complex financial data, making it more approachable for retail investors.
- **Scalable and Flexible Architecture**: Easily adaptable to additional data sources or use cases, ensuring long-term usability.

Limitations and Improvements

Limitations:

- Limited Dataset for Testing: The system has been tested with a relatively small dataset. Its performance and adaptability to larger and more complex databases remain uncertain without additional testing.
- Compatibility Across Databases: The project is tailored for a specific MySQL-based schema, and further development is needed to ensure compatibility with other database systems and varied data formats.
- Complex Query Scenarios: Minimal testing has been conducted for highly complex or edge-case queries. The system's stability and accuracy under significant input stress are yet to be fully validated.

Suggested Improvements:

- Expanded Testing: Increase the scale and diversity of datasets to better simulate real-world usage and identify potential performance bottlenecks.
- Database and Format Adaptation: Develop modules to handle a broader range of database systems (e.g., PostgreSQL, SQLite) and data formats (e.g., JSON, XML).
- Stress Testing: Conduct rigorous stress testing to evaluate system stability with large numbers of concurrent users and complex queries.
- Performance Optimization: Optimize query generation and execution time to ensure efficiency, even under high-demand scenarios.

Outputs and Instructions

Please see github page to view the results and instructions to deploy

Attachment

See next page for Phase I.

Team members: Tianyu Shi, Boyang Lyu
Target domain: Financial and ESG (Environmental, Social, Governance) analysis of Chinese home appliance companies
English question:

1. Find companies with more than 1000 employees, a registered capital exceeding 100 million, total comprehensive income attributable to the parent company over 50 million in 2022, and ESG environment and social scores above 80 and 85, respectively.
2. Rank companies in the "Home Appliances" sector by their average Return on Equity (ROE) and list the top 5 companies along with their average ROE.
3. Identify companies that went public after 2010, have an average inventory turnover exceeding 5 days, and have a financial leverage (debt-to-equity ratio) below 0.5.
4. List companies whose total assets to total liabilities ratio is in the top 10% among all companies, along with their company names and ratios.
5. Find the top 5 companies with the highest total operating revenue in each industry sector, along with their industry name.
6. Identify companies with a year-over-year growth rate in operating revenue exceeding 20% in the past three years, grouped by their industry and sorted by their growth rate.
7. Compute the cumulative R&D expenses for all companies in the "Information Technology" sector over the past 5 years, and list the top 3 contributors.
8. Find companies with registered capital above $50 million and total liabilities exceeding total assets in any year, and return their names and respective years.
9. List all companies whose IPO dates are earlier than their establishment dates due to data errors, along with the erroneous data.
10. Identify companies with at least one ESG score (environmental, social, governance) missing or null, and return their company names and the missing fields.
11. Compute the correlation between total assets and net profit for companies in the financial sector and display the correlation coefficient.
12. For companies listed on the Shenzhen Stock Exchange, calculate the average financial leverage (debt-to-equity ratio) by industry and identify the industry with the lowest average.
13. Track the trend of ESG environment scores for companies with the highest year-over-year growth in profit over the past 3 years.
14. List all companies with registered capital exceeding $50 million and compute their average employee count grouped by their industries.
15. Find companies whose total operating costs exceed 90% of their operating revenue for the last three years, and return their names and profit margins.

16. Identify companies whose average quick ratio is below 1.0 over the past 5 years and provide recommendations for improvement based on their financial data.
17. Find companies that achieved an average inventory turnover ratio above the industry median in the "Consumer Goods" sector and list their names and ratios.
18. Compare the financial performance of companies in the "Renewable Energy" and "Traditional Energy" sectors by computing the average return on equity, profit margin, and R&D expenses for the last 3 years.

Relational Data Model and Schema

```
CREATE TABLE Securities (
   `Securities_Code` VARCHAR(20) NOT NULL, -- Stock code, usually a string
   `sec_englishname` VARCHAR(100),         -- Company English name, string
   `ipo_date` DATE,                        -- IPO date, date type
   `exch_eng` VARCHAR(10),                 -- Exchange code, string
   `country` VARCHAR(10),                  -- Country code, string
   PRIMARY KEY (`Securities_Code`)         -- Primary key
);

CREATE TABLE Companies (
   `Company_code` VARCHAR(10) NOT NULL,       -- Company code, string, primary key
   `Securities_Code` VARCHAR(20),             -- Stock code, string
   `comp_name_eng` VARCHAR(255),              -- Company English name, string
   `phone` TEXT,                              -- Phone number(s), may contain multiple numbers, stored as TEXT
   `industry_swcode_2021` BIGINT,             -- Industry code, uses large integer type
   `founddate1` DATE,                         -- Establishment date, using DATE type
   `regcapital` DECIMAL(15, 4),               -- Registered capital, accurate to 4 decimal places
   `employee_number` INT,                     -- Number of employees, integer type
   PRIMARY KEY (`Company_code`),              -- Primary key
   FOREIGN KEY (`Securities_Code`) REFERENCES Securities(`Securities_Code`) -- Foreign key linking to Securities
);

CREATE TABLE balance_sheet (
   `balanceCode` VARCHAR(10) NOT NULL,        -- Balance sheet code
   `windcode` VARCHAR(20),                    -- Stock code
   `tot_assets` DECIMAL(15, 4),               -- Total assets
   `tot_cur_assets` DECIMAL(15, 4),           -- Current assets
   `inventories` DECIMAL(15, 4),              -- Inventories
   `fix_assets` DECIMAL(15, 4),               -- Fixed assets
```

```
    `tot_liab` DECIMAL(15, 4),                    -- Total liabilities
    `tot_non_cur_liab` DECIMAL(15, 4),            -- Non-current liabilities
    `tot_cur_liab` DECIMAL(15, 4),                -- Current liabilities
    `eqy_belongto_parcomsh` DECIMAL(15, 4),       -- Equity attributable to
shareholders of the parent company
    PRIMARY KEY (`balanceCode`),                  -- Primary key
    FOREIGN KEY (`windcode`) REFERENCES Securities(`Securities_Code`) --
Foreign key linking to Securities
);

CREATE TABLE income_statement (
    `IncomeCode` VARCHAR(10) NOT NULL,            -- Income statement code
    `BalanceCode` VARCHAR(10) NOT NULL,           -- Balance sheet code
    `tot_oper_rev` DECIMAL(15, 4),                -- Total operating revenue
    `tot_oper_cost` DECIMAL(15, 4),               -- Total operating costs
    `opprofit` DECIMAL(15, 4),                    -- Operating profit
    `tot_profit` DECIMAL(15, 4),                  -- Total profit
    `tax` DECIMAL(15, 4),                         -- Taxes
    `net_profit_is` DECIMAL(15, 4),               -- Net profit
    `tot_compreh_inc_parent_comp` DECIMAL(15, 4),  -- Comprehensive income
attributable to the parent company
    `selling_dist_exp` DECIMAL(15, 4),            -- Selling expenses
    `gerl_admin_exp` DECIMAL(15, 4),              -- General and administrative
expenses
    `rd_exp` DECIMAL(15, 4),                      -- R&D expenses
    `fin_exp_is` DECIMAL(15, 4),                  -- Financial expenses
    PRIMARY KEY (`IncomeCode`),                   -- Primary key
    FOREIGN KEY (`BalanceCode`) REFERENCES balance_sheet(`balanceCode`)
-- Foreign key linking to balance_sheet
);

CREATE TABLE financial_analysis (
    `FAID` VARCHAR(10) NOT NULL,                  -- Financial analysis table code
    `BalanceCode` VARCHAR(10) NOT NULL,           -- Balance sheet code
    `eps_basic` DECIMAL(15, 4),                   -- Basic earnings per share
    `roe_basic` DECIMAL(15, 4),                   -- Return on equity (basic)
    `profittogr` DECIMAL(15, 4),                  -- Total profit growth rate
    `currentRatio` DECIMAL(15, 4),                -- Current ratio
    `quick` DECIMAL(15, 4),                       -- Quick ratio
    `fa_debttoeqy` DECIMAL(15, 4),                -- Financial leverage (debt-to-equity ratio)
    `invturndays` DECIMAL(15, 4),                 -- Inventory turnover days
    `invturn` DECIMAL(15, 4),                     -- Inventory turnover rate
    `turnover_ttm` DECIMAL(15, 4),                -- Turnover (TTM)
```

```
    `yoy_tr` DECIMAL(15, 4),                -- Year-over-year operating revenue growth
rate
    `yoyprofit` DECIMAL(15, 4),             -- Year-over-year profit growth rate
    PRIMARY KEY (`FAID`),                   -- Primary key
    FOREIGN KEY (`BalanceCode`) REFERENCES balance_sheet(`balanceCode`)
-- Foreign key linking to balance_sheet
);

CREATE TABLE esg_data (
    `ESG_reportID` VARCHAR(10) NOT NULL,          -- ESG report ID
    `Company_code` VARCHAR(10) NOT NULL,          -- Company code
    `esg_rating_wind` VARCHAR(10),                -- ESG rating (e.g., AA, BBB, etc.)
    `esg_escore_wind` DECIMAL(10, 2),             -- ESG environment score
    `esg_sscore_wind` DECIMAL(10, 2),             -- ESG social score
    `esg_gscore_wind` DECIMAL(10, 2),             -- ESG governance score
    `esg_rating_ftserussell` DECIMAL(10, 2),      -- FTSE Russell ESG rating (nullable)
    `esg_mgmtscore_wind2` DECIMAL(10, 3),         -- ESG management score
    `esg_eventscore_wind2` DECIMAL(10, 3),        -- ESG event score
    PRIMARY KEY (`ESG_reportID`),                 -- Primary key
    FOREIGN KEY (`Company_code`) REFERENCES Companies(`Company_code`)
-- Foreign key linking to Companies
);
```

Plan

All our financial data is sourced from the Wind financial database and its quantitative API. Utilizing professionally curated statistics provided by a trusted financial data company, we built and analyzed a MySQL database tailored to the financial data of specific industries.

Wind is a leading financial data platform offering comprehensive, accurate, and timely data on equities, bonds, macroeconomics, and more. Widely used by financial institutions and researchers, it provides powerful APIs for efficient data extraction and analysis, making it an essential tool for financial and economic research.

Form/Type of output

We utilize Python, along with the Django web framework, to render the user interface (UI) for our application. This approach allows us to efficiently manage backend operations and seamlessly integrate them with the frontend. Django's robust templating engine enables us to dynamically generate HTML, ensuring that the UI is both responsive and interactive. By leveraging Django's features, such as context rendering, reusable templates, and form handling, we can deliver a polished and user-friendly interface that aligns with the application's functionality and design

requirements. This setup also supports scalable development and easy customization to meet evolving user needs.

specialized/advanced topics
Our focus is primarily on natural language interfaces that leverage Large Language Models (LLMs) to seamlessly translate natural language inputs into SQL queries.

Additionally, we aim to provide a visually appealing and user-friendly GUI, complemented by the capability to generate reports in Excel format, enabling users to download them instantly for their convenience.