# System Introduction

I am an IT guy who previously worked in a securities company, primarily involved in data processing, project management, and software development. Due to work requirements, I often work with SQL and Python languages, and by chance, I discovered that Python can not only process and analyze data but also develop web applications. This sparked my interest, leading me to develop the finance-calculator system in my spare time for learning purposes and to review my daily work content.

The finance-calculator system is a simple web application used for storing, processing, analyzing, and measuring risk indicators of stock holdings. It achieves data visualization through three basic processing steps: collection, measurement, and tabulation. In terms of technology selection, the front-end of the system is built using the React framework, the back-end is developed using the Flask framework in Python, and the database used is the open-source PostgreSQL database. The system is still under continuous development and iteration. Currently implemented functionalities include data collection, manipulation, statistical analysis, and VaR (Value at Risk) measurement. Below, I will provide detailed descriptions of these functionalities.

## Data Collection, Manipulation, and Editing

**Data Collection**

The system supports two modes of data acquisition: automatic collection and manual import. These correspond to the "ETL Task" and "Data Import" functionalities, respectively. Automatic collection is implemented using Python's BackgroundScheduler, which runs at a fixed time each day to retrieve data from specific Excel files in a designated folder. It collects data from the day before the program's execution date and automatically deletes corresponding data from the database before collection to ensure data uniqueness. Manual import serves as a supplement to automatic collection, allowing users to import historical data or data that cannot be collected automatically. Before importing data, the system checks for date conflicts with existing data to ensure successful importation.

During data collection and importation, the system automatically calls the "BaoStock" API to retrieve closing prices and industry information for securities held on the specified dates, calculates the market value of securities, and assigns industry labels to them.

**Data Manipulation (Add, Delete, Edit)**

The "Data Editing" functionality is used to manipulate imported data. If there are errors in the automatically collected or manually imported data, this feature allows users to rectify them. Users can filter data based on various criteria and modify or delete selected data to correct errors in data collection or importation.

## **Statistical Analysis of Data**

The system supports two modes of data analysis: detailed data analysis and summary data analysis, corresponding to the "holdingshow" and "holdinganalysis" functionalities, respectively. In "holdingsho," detailed data is analyzed based on date, department, investment portfolio, industry, and security code. In "holdinganalysis," users can choose to analyze data at the "Company," "Department," or "Portfolio" level. Python's pandas library is mainly used for data processing to generate summarized data, including information on the top ten stocks by market value, daily market value summaries for the same year as the query date, concentration of market value on the query date, and industry-wise market value concentration on the query date. These summarized data are sent to the front-end in JSON format, and React on the front-end uses the react-chartjs-2 component to draw corresponding line charts and pie charts based on the data.

## **Measurement of VaR (Value at Risk)**

This part corresponds to the "VaR Measurement" functionality in the system. It supports VaR measurement using three methods: historical simulation, parametric method, and Monte Carlo simulation. Users can freely select confidence level, forecast days, length of historical data, and in the case of Monte Carlo simulation, the number of simulation paths. After measurement, the results are displayed in a drill-down table and as line charts and frequency distribution histograms. The operational principles of this functionality are as follows:

**Measurement Process:**

Retrieve corresponding holding data from the database based on the date parameters passed from the front-end.

Calculate the weight of each stock's market value in the investment portfolio, department, and company.

Call the "BaoStock" API to obtain the closing prices of each stock for each day in the selected historical date range and calculate the logarithmic returns for each stock.

Combine the returns with the previously calculated weight data to calculate the logarithmic returns for the investment portfolio, department, and company for each day.

Perform VaR measurement based on the selected method:

Historical method: Divide the returns of the corresponding level based on quantiles to obtain the VaR at the specified confidence level.

Parametric method: Calculate the mean and standard deviation of the returns for the corresponding level, and use a predefined formula to calculate the VaR at the specified confidence level.

Monte Carlo simulation method: Calculate the mean and standard deviation of the returns for the corresponding level, then simulate random paths based on the input number of paths. Divide the simulated returns based on quantiles to obtain the VaR at the specified confidence level.

**Drill-Down Table:**

Combine VaR at different levels into a large table containing company, department, investment portfolio, and individual securities. This data is transmitted to the front-end in JSON format, and React on the front-end implements drill-down display of data through recursive methods.

**Charts:**

Pass necessary parameters to the front-end in JSON format, and React on the front-end uses react-chartjs-2 and chart.js to draw frequency distribution and normal distribution charts based on this data.

## Conclusion

The finance-calculator system is developed during spare time, and there is still significant room for optimization due to limitations in my technical skills, business knowledge, and time constraints. For example, VaR algorithms can consider correlation issues, expand securities types to include bonds and derivatives, support more databases for automatic collection, improve the aesthetics of the front-end pages, and consider efficiency issues in the back-end architecture. The main purpose of open-sourcing this system is to stimulate discussion and exchange of ideas with others. I will continue to maintain and improve the system as time allows.