🐂Finance Bull Meeting Notes

1/28:

Today, we reviewed the lab assignment and agreed that each person should read through the entire assignment requirements and consider what to include.
Additionally, everyone should work on installing the packages mentioned in the PDF file as part of today's progress.

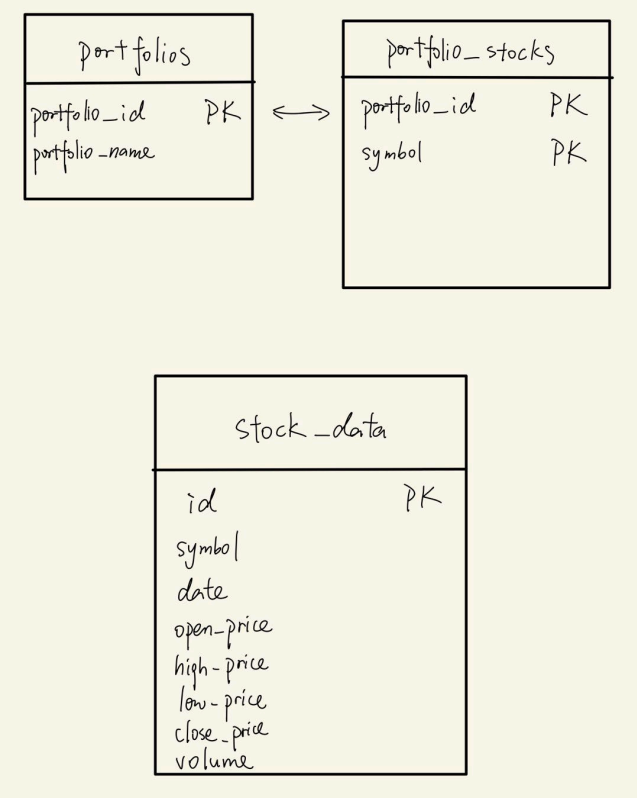
1/29:

We divided the lab into 3 parts: fetching data, designing portfolio operations, and data preprocessing. And each person is responsible for each part.
For the Data Fetching part: we decided the following. And we'll store each stock's data into a table using SQL in python.
Stock to fetch: Tesla, Apple, Nvidia, xxxxx.
Input data range: 2020/01/01 → 2024/12/31

1/30:

Database Design:

We originally planned to store data for each stock in a separate table, but this is a common misconception. Imagine managing 500+ stocks—this would require 500 table definitions, inserts, indexes, and more, making maintenance extremely complicated. Additionally, combining data across multiple symbols allows for easier aggregation and analysis. Therefore instead, we developed another database design (see the ER diagram below).



We'll have 3 tables in our stock_analysis database:

- Portfolios: different investment portfolios. Eg. Portfolio 1 is named "Tech Giants". Portfolio 2 is named "Pharma Picks".
- Portfolio_stocks: the stocks that belong to each portfolio. Eg. Apple and Google would belong to Tech Giants.
- Stock_data: historical stock price data for different stocks.

1/31:

We store the processed data metrics in a single table called *daily_metrics*, which includes the following columns: *symbol, date, daily_return, cumulative_return, SMA_10*(simple moving average in 10 days), and *volatility_20*. The primary key for this table is a combination of *symbol* and *date*. This structure allows us to conveniently calculate various metrics for the portfolio without complicating matters by mixing them with the *stock_data* table. The metrics we have chosen are recorded on a daily basis, and if needed, we can easily add more metrics at different intervals in the future using our code.

Additionally, we discussed the progress of our lab to ensure everything is on track today.

2/1:

We designed an automated stock trading simulator that allows users to simulate trades based on historical stock data stored in the database. Users can input their initial investment, select a stock, define a date range, and choose a trading strategy. The system retrieves historical prices, executes buy and sell orders based on the chosen strategy, and calculates the final portfolio value.

The script supports multiple trading strategies, such as momentum-based trading, moving average crossovers, and threshold-based buying/selling. After executing the trades, it logs all transactions and prints the final portfolio value along with the return on investment (ROI).

This system provides a practical way to evaluate different trading strategies using real stock data, helping users analyze potential profitability before applying them to live markets.