Stock Span Tracker

1. Introduction

Stock market investors face challenges in quickly analyzing trends and making decisions. The Stock Span Problem is a well-known problem in financial data analysis that helps investors determine how long a stock price has been rising compared to previous days. We developed a Stock Span Tracker system that efficiently computes and stores span values of stock prices using database concepts. And we call it the STOCKER

2. Problem Overview

The Stock Span Tracker project aims to provide an efficient system for analyzing stock market trends by storing, managing, and calculating stock span data for various companies. The system helps users and analysts retrieve meaningful insights such as trends, comparisons, and personalized portfolio management.

Key Features

Store Stock Prices – Stores the daily stock prices of different companies.

Calculate Span – Determines the maximum consecutive days where the stock price is ≤ today’s price.

Efficient Queries – Retrieve spans, price trends, and company comparisons quickly.

User Portfolio Management – Users can add companies to their personal watchlist/portfolio

3. scope of the project

The Stock Span Tracker (Stocker) project is designed to provide a complete solution for stock price analysis, span calculation, and portfolio management.

In Scope:

Maintaining a database of daily stock prices of companies.
Implementing an algorithm to calculate stock span efficiently.
Supporting real-time calculation for newly added prices.
Efficient queries for trend retrieval and comparisons.
Portfolio & watchlist management for users.
Scalable design to handle multiple companies over time.

4. Objectives of the Project

The primary objective of the Stock Span Tracker (Stocker) project is to design and develop a system that efficiently computes and manages stock span values, stores stock data, and provides meaningful insights to investors and traders.

Specific Objectives:

Stock Data Management
To systematically collect and store the daily stock prices of multiple companies.
To maintain historical stock records for long-term analysis and retrieval.

Span Calculation
To implement the Stock Span algorithm that calculates the number of consecutive days the stock price has been less than or equal to the current day’s price.
To optimize the calculation process for efficiency, ensuring faster computation even for large datasets.

Efficient Query Processing
To enable users to perform quick queries to retrieve span values, stock trends, and comparisons.
To ensure database-backed operations that provide accurate and consistent results.

User Portfolio and Watchlist Management
To allow users to create personalized watchlists of selected companies.
To provide a dashboard for monitoring the performance of stocks in their portfolio.

Decision Support for Investors and Traders
To provide insights into stock strength, enabling investors to evaluate current price performance against past data.
To help traders identify potential buying and selling opportunities through span trends.

Scalability and Flexibility
To design a system that can track multiple companies simultaneously.
To ensure the solution works effectively for both small datasets (few companies) and large datasets (many companies over long periods).

User-Friendly System Design
To develop an interface that is simple, intuitive, and easy for non-technical users.
To present stock data and span analysis results in a clear, well-organized, and visual manner.

Foundation for Future Enhancements
To create a robust base system that can later be extended with advanced features such as real-time stock data integration, predictive analytics, or AI-driven recommendations.

5. **Security and Access Control**
   Protect sensitive information through proper database security measures and role-based access.
6. **Scalability and Maintainability**
   Ensure the database can handle future growth and can be easily updated without disrupting operations.
7. **Support for Reporting and Analytics**
   Provide the ability to generate reports for:
   - Customer account details.
   - Loan repayment tracking.
   - Transaction summaries.
   - Branch and employee performance.
8. **Minimizing Redundancy**
   Apply normalization techniques to reduce data duplication, optimize storage, and maintain consistency.

5. Significance of the Project

The Stock Span Tracker (Stocker) holds significant value for both investors and traders by providing a structured way to analyze stock prices and trends. It bridges the gap between raw financial data and meaningful insights that support decision-making.

Key Significance:

Quick Analysis
Provides investors with a fast and reliable way to evaluate how strong the current stock price is compared to past prices.
Saves time in manually analyzing historical stock movements.

Decision Support
Helps traders and investors identify buying and selling opportunities by highlighting short-term price trends.
Assists in making informed decisions rather than relying only on guesswork or intuition.

Database-Backed System
Ensures that stock data is systematically stored, retrieved, and updated.
Reduces the risk of data loss and improves accuracy in financial analysis.

Scalability
Can track multiple companies’ stock performances over long periods of time.
Supports growth from small datasets to large-scale stock portfolios.

Practical Relevance
Provides a real-world application of data structures, algorithms, and database concepts.
Offers students, researchers, and professionals a tool that demonstrates the integration of finance with computer science concepts.

Foundation for Future Enhancements
Acts as a base system that can be extended with advanced features like real-time market integration, predictive modeling, or AI-driven recommendations.

Entities and their Attributes

User
user_id (PK)
username
email
registration_date

Portfolio
portfolio_id (PK)
user_id (FK)
creation_date
invested_amount

Watchlist
watchlist_id (PK)
user_id (FK)
stock_id (FK)
date_added

Alerts
alert_id (PK)
stock_id (FK)
trigger_price
alert_type
status

Stocks
stocks_id (PK)
symbol
company_name
sector

Price
price_id (PK)
stock_id (FK)
date
open_price
close_price
high_price

Transaction
transaction_id (PK)
portfolio_id (FK)
stocks_id (FK)
brokerage_id (FK)
quantity
transaction_date

Ratings
rating_id (PK)
stock_id (FK)
rating_name
rating

Brokerage
brokerage_id (PK)
name
contact_no
license_no

8. Relationships

The entities in the Stock Span Tracker (Stocker) are interrelated to represent how users, stocks, portfolios, transactions, and brokers interact within the system. The relationships between entities are defined as follows:

User → Watchlist
One-to-Many: A user can maintain multiple watchlists containing different stocks.

User → Portfolio
One-to-Many: A single user can create and manage multiple portfolios.

User → Alerts
One-to-Many: A user can set multiple alerts for different stocks based on conditions.

Portfolio → Transaction
One-to-Many: A single portfolio can execute multiple transactions.

Transaction → Broker
Many-to-One: Multiple transactions can be executed through a single broker.

Stocks → Watchlist
One-to-Many: A stock can appear in multiple user watchlists.

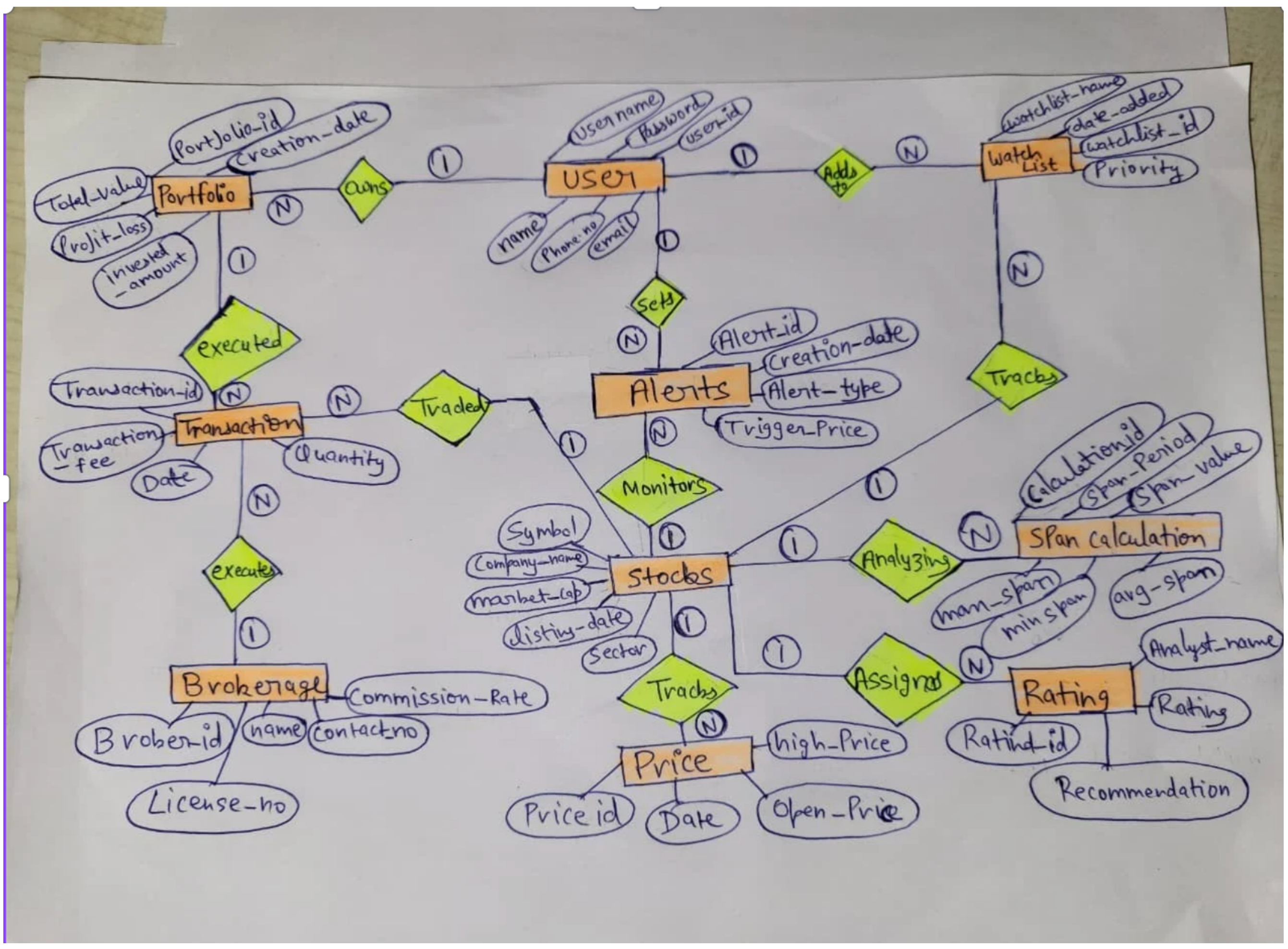
Stocks → Alerts
One-to-Many: A stock can have multiple alerts based on various trigger conditions.

Stocks → Ratings
One-to-Many: A stock can receive multiple ratings from different rating sources.

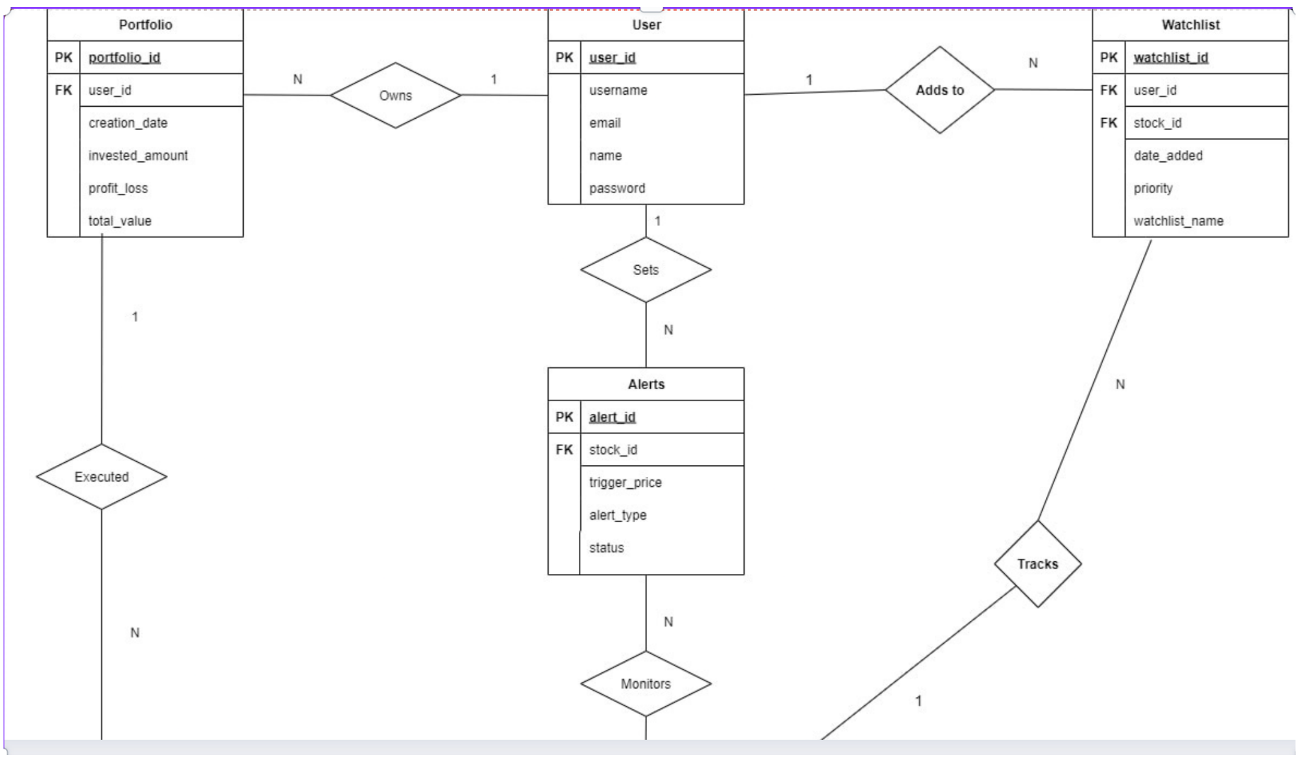
Stocks → Price
One-to-Many: A stock can have multiple price records (open, close, high) over time.

Stocks → Transaction
One-to-Many: A stock can be traded multiple times across different transactions.

ER daigram



ER model



ER to Relational Model (Schema)

User(user_id PK, username, email, name, password)

Portfolio(portfolio_id PK, user_id FK, creation_date, invested_amount, profit_loss, total_value)

Watchlist(watchlist_id PK, user_id FK, stock_id FK, date_added, priority, watchlist_name)

Alerts(alert_id PK, stock_id FK, trigger_price, alert_type, status)

Stocks(stock_id PK, symbol, company_name, sector, listing_date)

Transaction(transaction_id PK, portfolio_id FK, stock_id FK, brokerage_id FK, quantity, transaction_date)

Brokerage(brokerage_id PK, name, contact_no, license_no, commission_rate)

Price(price_id PK, stock_id FK, date, open_price, high_price)

Span_Calculation(calculation_id PK, stock_id FK, span_period, span_value, avg_span, min_span, max_span)

Ratings(rating_id PK, stock_id FK, rating, recommendation, analyst_name)

Queries (Relational Algebra)

Q1 List all usernames and emails of users.
→ π username, email (User)

Q2 Find all stock symbols and their sectors.
→ π symbol, sector (Stocks)

Q3 Get portfolios where invested amount is greater than 1,00,000.
→ σ invested_amount > 100000 (Portfolio)

Q4 Retrieve all watchlists that have High priority.
→ σ priority = 'High' (Watchlist)

Q5 Display names and commission rates of all brokerages.
→ π name, commission_rate (Brokerage)

Q6 Find all users with their portfolios.
→ User ⋈ User.user_id = Portfolio.user_id (Portfolio)

Q7 List portfolios along with their transactions.
→ Portfolio ⋈ Portfolio.portfolio_id = Transaction.portfolio_id (Transaction)

Q8 Show transactions along with stock details.
→ Stocks ⋈ Stocks.stock_id = Transaction.stock_id (Transaction)

Q9 Get users and their watchlists.
→ User ⋈ User.user_id = Watchlist.user_id (Watchlist)

Q10 Find stock prices with stock details.
→ Stocks ⋈ Stocks.stock_id = Price.stock_id (Price)

Q11 Find transactions executed by brokerages with commission rate < 0.02.
→ σ commission_rate < 0.02 (Brokerage) ⋈ Transaction

Q12 List users who have a profit in their portfolio.
→ σ profit_loss > 0 (Portfolio) ⋈ User

Q13 Find watchlists that contain IT sector stocks.
→ σ sector = 'IT' (Stocks) ⋈ Watchlist

Q14 Get all stocks recommended as Buy.
→ σ recommendation = 'Buy' (Ratings) ⋈ Stocks

Q15 Find stocks that have active Price Drop alerts.
→ σ alert_type = 'Price Drop' (Alerts) ⋈ Stocks

Q16 List usernames of users and the stocks they have traded.
→ π username, symbol (User ⋈ Portfolio ⋈ Transaction ⋈ Stocks)

Q17 Display stock company names along with their ratings.
→ π company_name, rating (Ratings ⋈ Stocks)

Q18 Show usernames, stock symbols, and transaction dates for all trades.
→ π name, symbol, transaction_date (User ⋈ Portfolio ⋈ Transaction ⋈ Stocks)

Q19 List stock IDs and their average span values.
→ π stock_id, avg_span (Span_Calculation)

Q20 Find watchlist names and stock symbols they track.
→ π watchlist_name, symbol (Watchlist ⋈ Stocks)

Q21 Find all stocks that are either in transactions or in watchlists.
→ π stock_id (Transaction) ∪ π stock_id (Watchlist)

Q22 Find stocks that are in watchlists but not traded yet.
→ π stock_id (Watchlist) − π stock_id (Transaction)

Q23 Rename the User relation as U.
→ ρ U (User)

Q24 List names that exist as users but not as brokerages.
→ π username (User) − π name (Brokerage)

Q25 Get a combined list of all analysts and broker names.
→ π analyst_name (Ratings) ∪ π name (Brokerage)