CHITKARA UNIVERSITY

DATABASE MANAGEMENT SYSTEM

STOCK SPAN TRACKER

TEAM -

Ayaan
2410990957

Anubhav
2410990933

Arshit
2410990947

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.

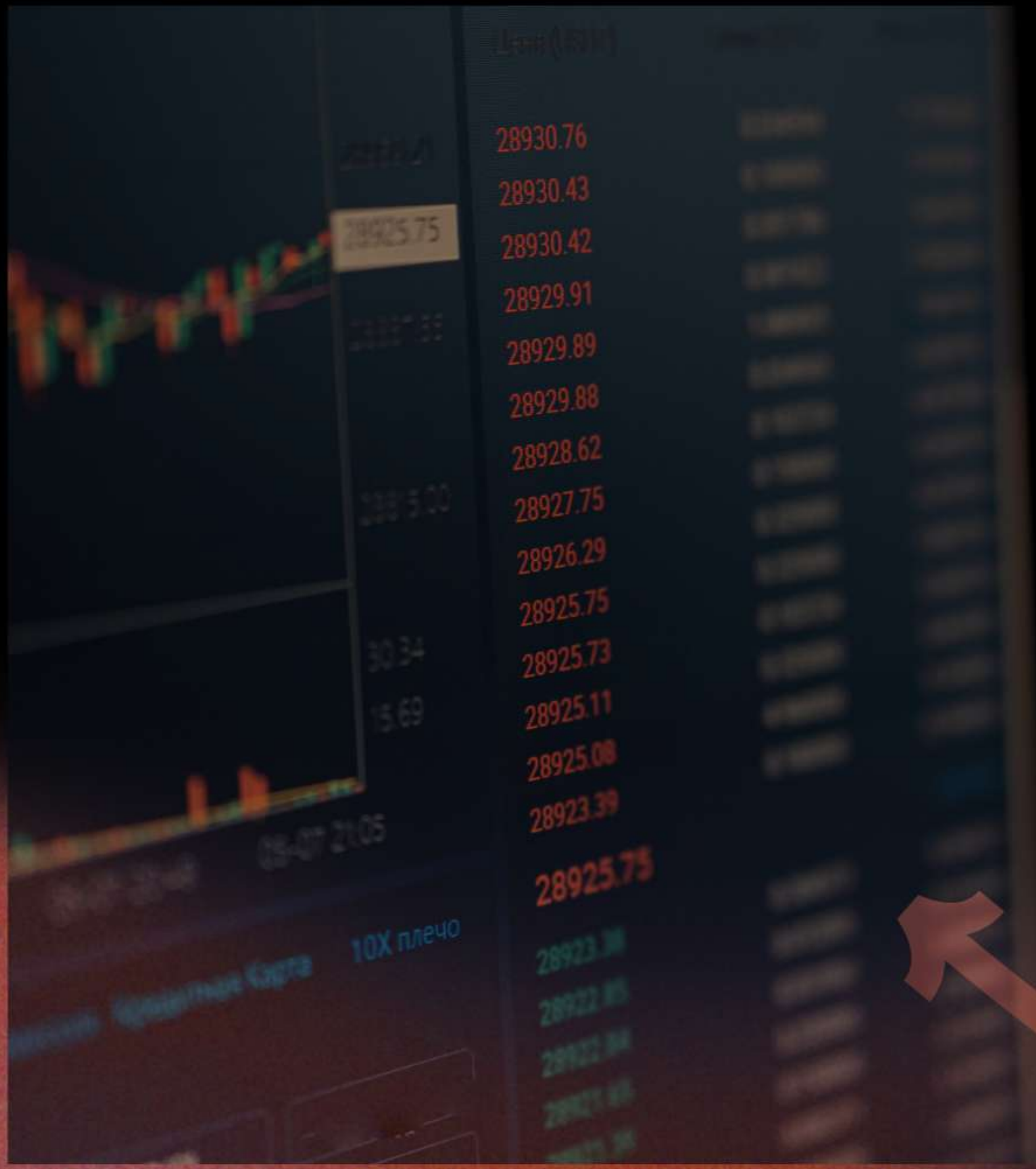
We developed a Stock Span Tracker system that efficiently computes and stores span values of stock prices using database concepts.

And we call it the STOCKER
~~STALKER~~

Project Overview

Stocker

Store Stock Prices	Daily prices of different companies.
Calculate Span	Determines the maximum consecutive days stock price ≤ today's price.
Efficient Queries	Retrieve span, trends, and comparisons easily.
User Portfolio Management	Users can add companies to their personal watchlist/portfolio.

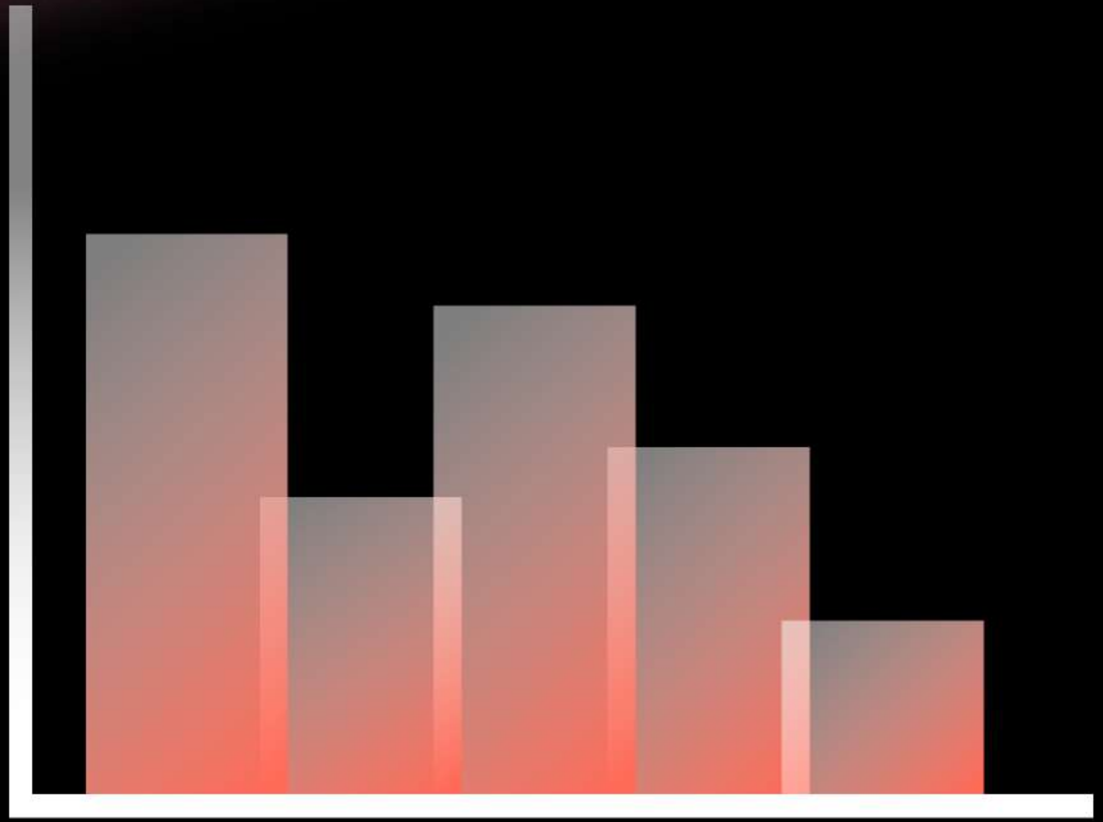


Problem Statement

Stock market investors often struggle to judge whether today's price rise is truly strong or just a one-day spike. For example, if a stock has been rising steadily for 5 days, that trend is more meaningful than a sudden jump after many down days.

The Stock Span Problem helps answer this by showing how many consecutive days before today the stock price was less than or equal to today's price. But calculating this manually across multiple companies is time-consuming and error-prone.

Existing platforms mostly show charts, which look informative but don't directly reveal this insight. To solve this gap, we created Stocker a Stock Span Tracker that stores data, calculates spans automatically, and provides quick, accurate insights for smarter decision-making.



Significance Of Stocker

01

Quick Analysis

Provides investors a fast way to know how strong the current price is compared to the past.

02

Decision Support

Helps traders identify buying/selling opportunities.

03

Database-Backed

Data is stored systematically for retrieval & updates.

04

Scalability

Can track multiple companies over long periods.

Entities And Attributes *Of Stocker*

01 User
user_id **(PK)**
username
email
registration_date

02 Portfolio
portfolio_id **(PK)**
user_id **(FK)**
creation_date
invested_amount

03 Watchlist
watchlist_id **(PK)**
user_id
stock_id
date_added

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

05 Stocks
stocks_id **(PK)**
symbol
company_name
sector

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

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

08 Ratings
rating_id **(PK)**
stock_id **(FK)**
rating_name
rating

09 Brokerage
brokerage_id **(PK)**
name
contact_no
license_no

Relationships

User to Watchlist
one to many (a user can have many stocks in watchlist section)

User to Portfolio
one to many (a user can have many portfolios)

Alert to User
many to one (one user can set many alerts for different stocks)

Portfolio to Transaction
one to many (one portfolio can execute many transactions)

Stocks to Watchlist
one to many (one stock can be there in many watchlists)

Transaction to Broker
many to one (many transactions can be executed by one broker)

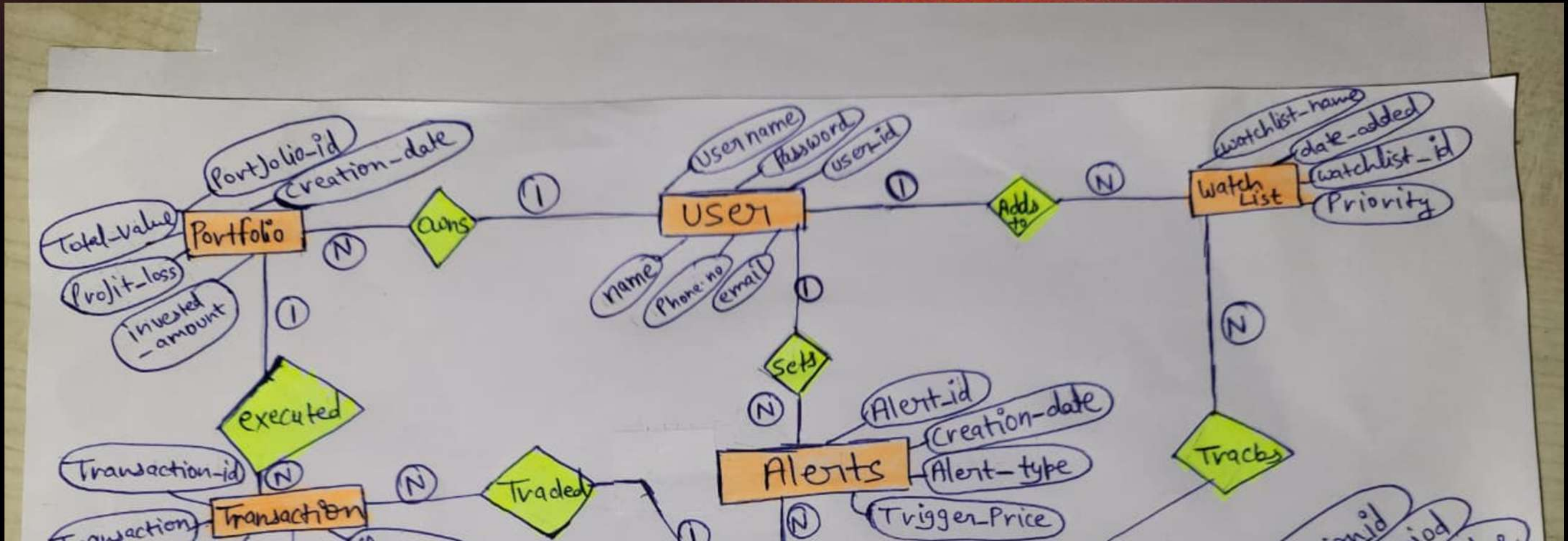
Stocks to Alert
one to many (one stock can have many alerts acc to its alert type)

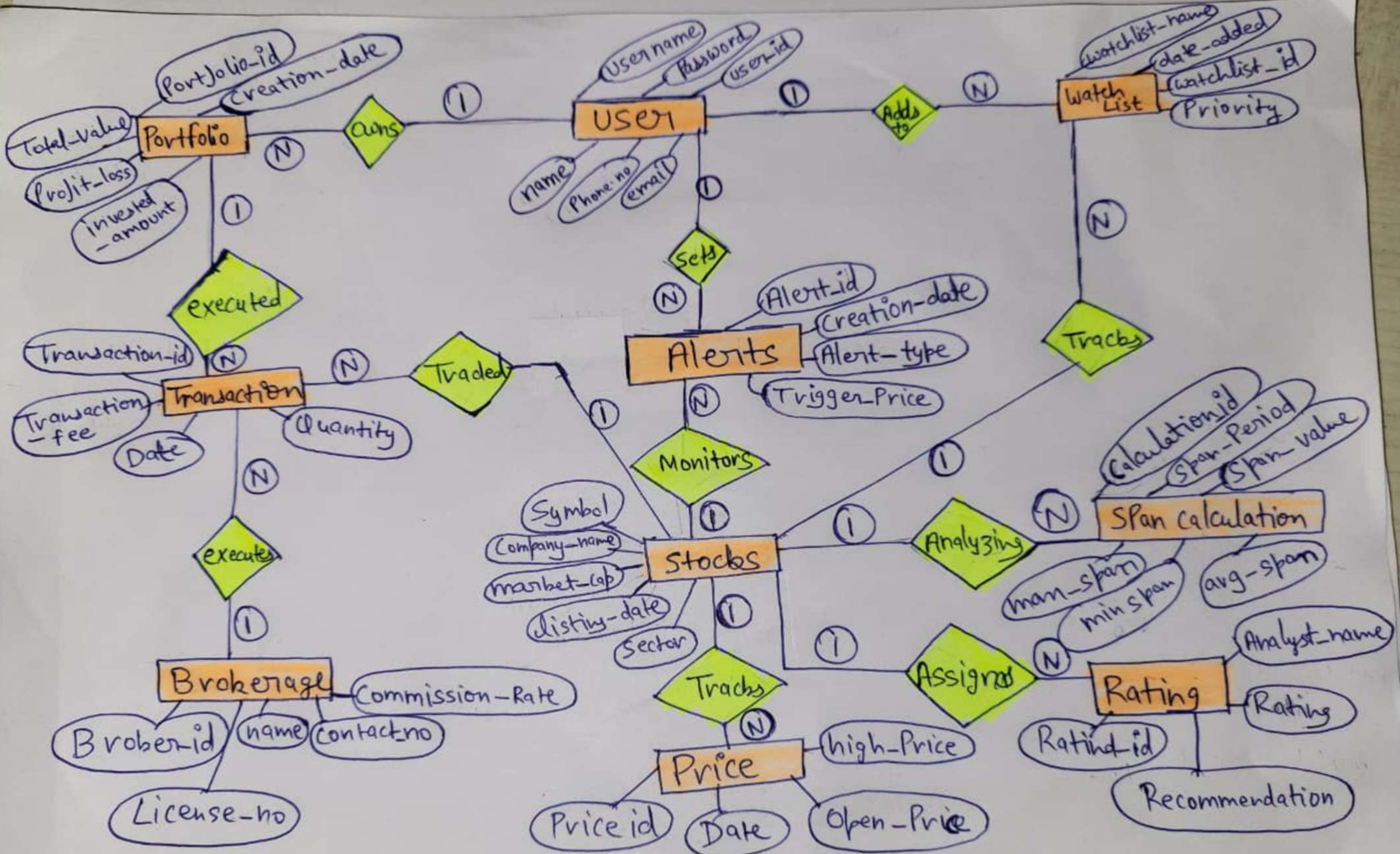
Stocks to Ratings
one to many (one stock can have many ratings from different rating id)

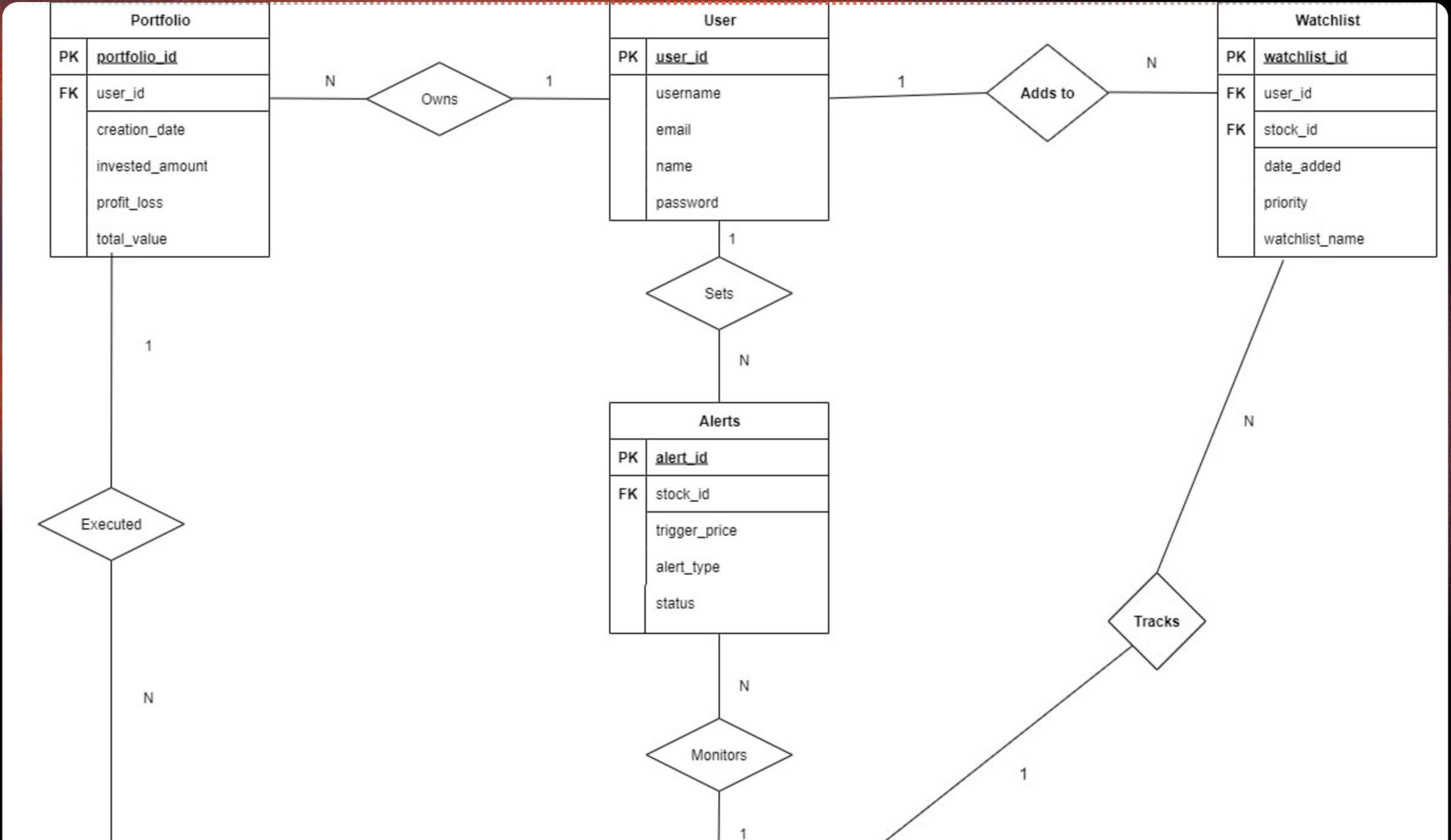
Stocks to Price
one to many (one stock may have different type of prices like open price , close price, high price)

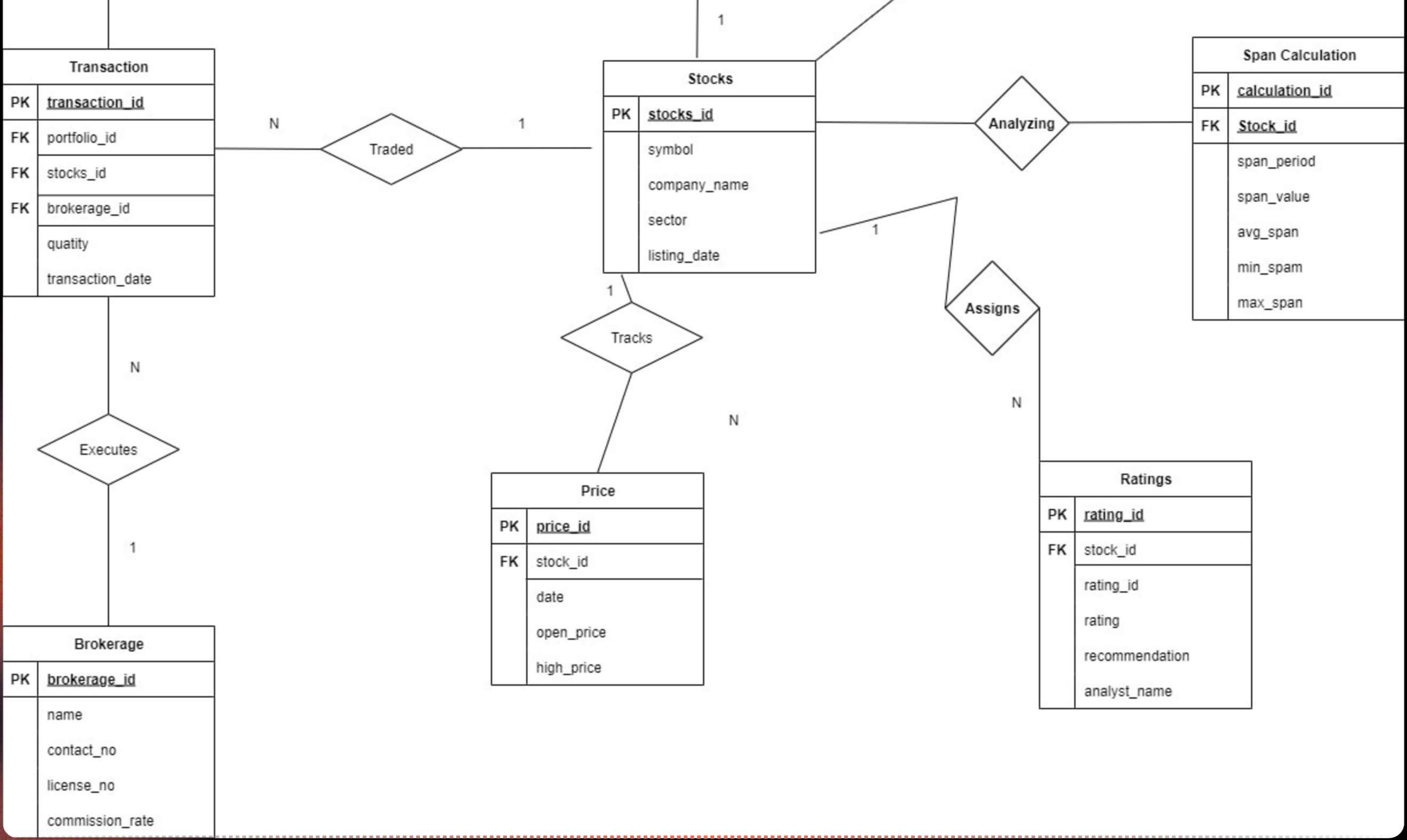
Stocks to Transaction
one to many (one stock can be traded multiple times)

ER Diagram


Portfolio-id
Creation-date
Total-Value
Portfolio
Profit-loss
Invested-amount
owns
1
N
User name
Password
user-id
User
name
Phone no
email
1
Adds to
N
watchlist-name
date-added
watchlist-id
Priority
Watch List
1
Sets
N
Alert-id
Creation-date
Alert-type
Trigger-Price
Alerts
N
Tracks
executed
1
N
Transaction-id
Transaction
N
Traded
1


Portfolio
Portfolio-id
creation-date
Total-value
Profit-loss
invested-amount
owns
1
N
User
User name
Password
user-id
name
Phone no
email
1
Adds to
N
Watch List
watchlist-name
date-added
watchlist-id
Priority
1
Sets
N
Alerts
Alert-id
creation-date
Alert-type
Trigger-Price
N
Tracks
N
executed
1
N
Transaction
Transaction-id
Transaction-fee
Date
Quantity
N
Traded
1
Monitors
N
1
N
executes
1
Brokerage
Commission-Rate
Broker-id
name
contact-no
License-no
Stocks
Symbol
Company-name
market-cap
listing-date
Sector
1
Analyzing
1
N
Span calculation
Calculation id
Span-Period
Span-value
max-span
min span
avg-span
1
Assigned
1
N
Rating
Analyst-name
Rating
Rating-id
Recommendation
1
Tracks
N
Price
high-Price
Price id
Date
Open-Price


Portfolio
PK portfolio_id
FK user_id
creation_date
invested_amount
profit_loss
total_value
N
Owns
1
User
PK user_id
username
email
name
password
1
Adds to
N
Watchlist
PK watchlist_id
FK user_id
FK stock_id
date_added
priority
watchlist_name
1
Sets
N
Alerts
PK alert_id
FK stock_id
trigger_price
alert_type
status
N
Monitors
1
1
Executed
N
N
Tracks
1



1
Transaction
PK transaction_id
FK portfolio_id
FK stocks_id
FK brokerage_id
quatity
transaction_date
N
Traded
1
Stocks
PK stocks_id
symbol
company_name
sector
listing_date
Analyzing
Span Calculation
PK calculation_id
FK Stock_id
span_period
span_value
avg_span
min_spam
max_span
1
Assigns
1
Tracks
N
N
N
Executes
1
Price
PK price_id
FK stock_id
date
open_price
high_price
Ratings
PK rating_id
FK stock_id
rating_id
rating
recommendation
analyst_name
Brokerage
PK brokerage_id
name
contact_no
license_no
commission_rate

ER To Relational Scheme

Stocker

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)

Queries (Relational Algebra)

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)

Queries (Relational Algebra)

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)

Conclusion

The Stock Span Tracker demonstrates how a classic stock market problem can be solved using database management concepts.
By storing stock data, calculating span values automatically, and allowing quick retrieval through queries, the system provides meaningful insights that help investors make smarter decisions.
This project not only shows the practical use of DBMS in finance, but also highlights how efficient data handling can simplify real-world problems.
In short, it bridges the gap between theory and application, turning a textbook algorithm into a useful financial tool.

TEAM ROCKET

THANK YOU !!