



# Investment Analytics

## Enhancing Portfolio Construction with NoSQL Databases

Team LNS

William Lei, Irene Na, Aparna Shankran

Summer 2024

Audience: A Peer Team of Data Scientists who are not familiar with NoSQL Database

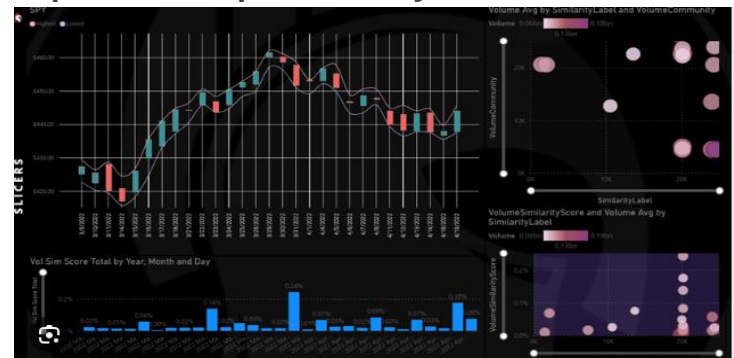
# Topic of Interest & Relevance

- In light of the data era, the hype about **discovering 'hidden patterns'** in the data has never been higher. **Real time analytics** are also increasingly in demand. It is especially so in quantitative trading world.
- Interestingly NoSQL databases tend to have edges in both areas.
- We are familiar with using relational database to analyze stock time series data, **could we utilize NoSQL databases to enhance portfolio construction?**
- Experimentation Ideas considered:
  - **Neo4j** algorithms to **discover hidden patterns** in order to enhance portfolio construction
  - **Redis and Mongodb** to enhance the **data timeliness**

# When Time Series Stock Data Meets



- Neo4j – As one of the most prolific NoSQL databases, Neo4j is a graph database management system. Unlike relational database, [Neo4j allows users to build, recognize and visualize more complex relationships among objects easily.](#)
- Big idea – Given stock market time series data, can we take advantage of Neo4j algorithms to enhance stock picking, portfolio optimization?
- The key approach: create a diversified portfolio, whose components are less correlated with each other, and are overall well performing
  - Key words: diversification; pick top performers
  - We experimented with several ideas to explore this possibility



# When Time Series Stock Data Meets neo4j (con.)

Build the graph– We used NASDAQ 100 data (from Apr to Sep 2021 as example), which include stock name, daily stock price, volume, market cap, P/E Ratio and industry

## Create 'stock' nodes

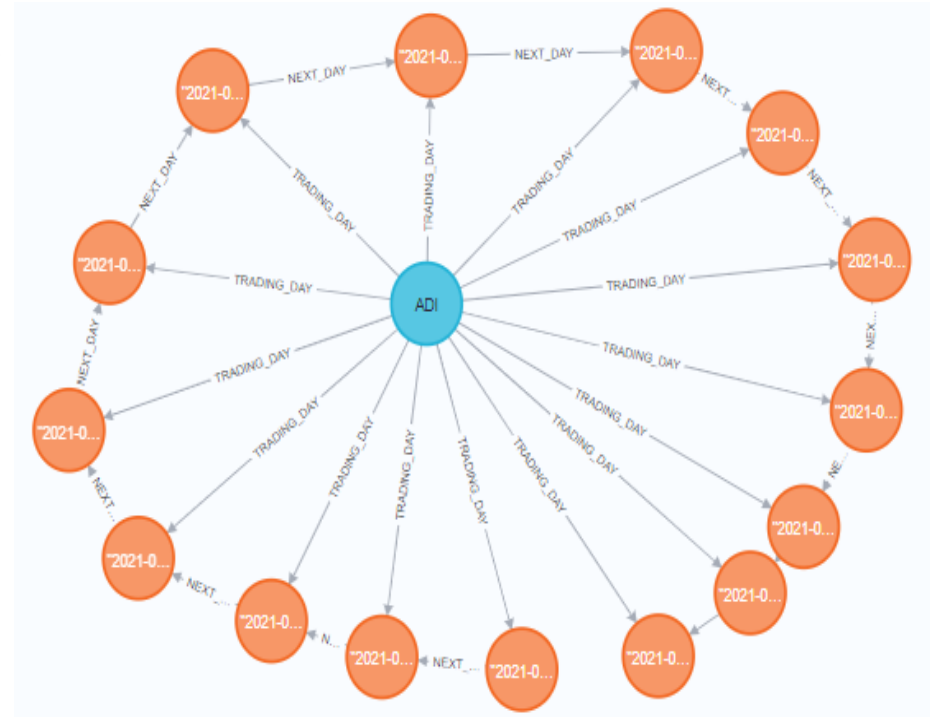
1. Create one 'stock' node for each unique ticker in NASDAQ-100

## Create 'day' nodes and 'trading\_day' relationship

1. Create 'day' node for each day, with properties include 'date', 'close' (price), 'volume', etc.
2. Build 'trading\_day' relationship between stock node and belong day nodes

## Create linked date list to connect day nodes for each stock node

1. Connecting day nodes belong to one stock sequentially based on 'date'
2. Set the sequence of close and volume as two stock properties, namely 'close\_array', and 'volume\_array'



# When Time Series Stock Data Meets neo4j (con.)

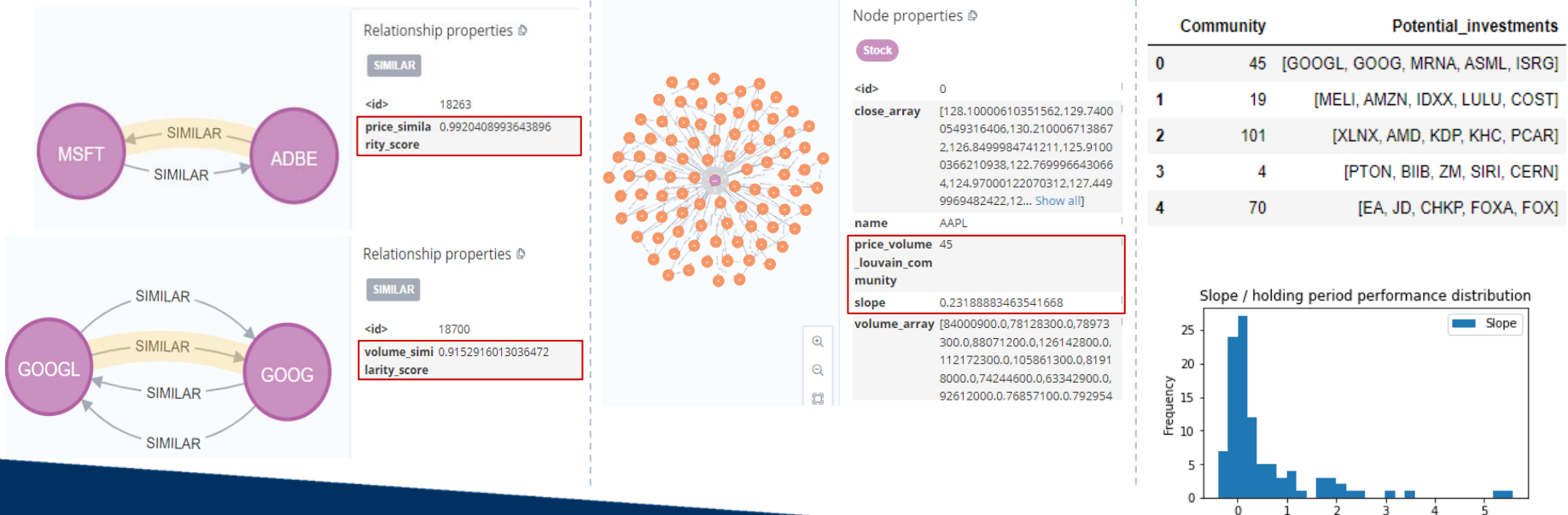
- Idea 1: Construct stock communities based on price and volume correlation similarities, then choose top performers from different communities as a portfolio, which maximize the return but minimize the risks

Build KNN similarity score based on price correlation for highly correlated stocks

Build KNN similarity score based on volume correlation for highly correlated stocks

Build Louvain community for stock nodes based on both similarity scores

Select top performers based on holding period performance in each community



# When Time Series Stock Data Meets neo4j (con.)

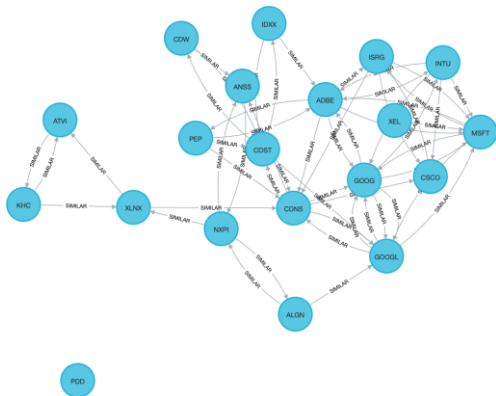
- Idea 2: Implement PageRank algorithm and identify stocks that are influential not just within their communities but across the entire Nasdaq 100, with the existing Price and Volume Correlation Similarity–Based Stock Communities

Determine PageRank scores for stocks using price and volume correlations

Rank top stocks globally

Rank top stocks within each community to identify influential stocks by PageRank

Leverage results for portfolio diversification and risk reduction

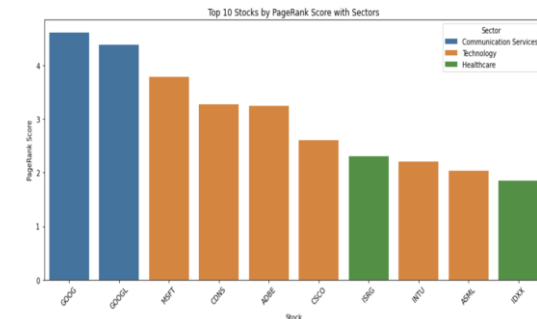


Top stock ranked by Page Rank

	Stock	score
0	GOOG	4.613370
1	GOOGL	4.381496
2	MSFT	3.776830
3	CDNS	3.275739
4	ADBE	3.236796
..	...	...
97	CTSH	0.193300
98	MNST	0.189899
99	REGN	0.150000
100	SBUX	0.150000
101	EBAY	0.150000

Top stock per Community

Top Stocks Per Community:			
Stock	Community	score	
40	FOX	4	0.469586
11	ANSS	25	1.597766
75	PCAR	67	1.770418
43	GOOG	81	4.613370
27	CSCO	98	2.600328



Goal: Identify stocks that are high influential to the broader market and also their respective sectors and communities to ensure diversification for our portfolio

# When Time Series Stock Data Meets neo4j (con.)

- Idea 3: Calculate Geodesic distance using market cap, PE Ratio and Industry – Euclidean distance is used as a proxy for ‘geographical’ distance

Calculate Euclidean distance between each pair of stocks

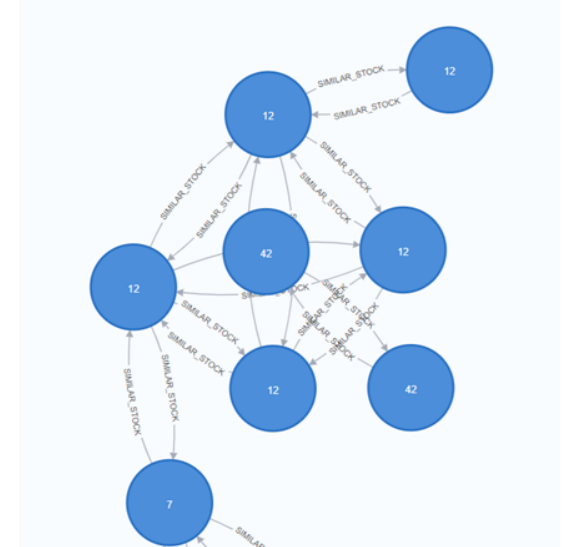
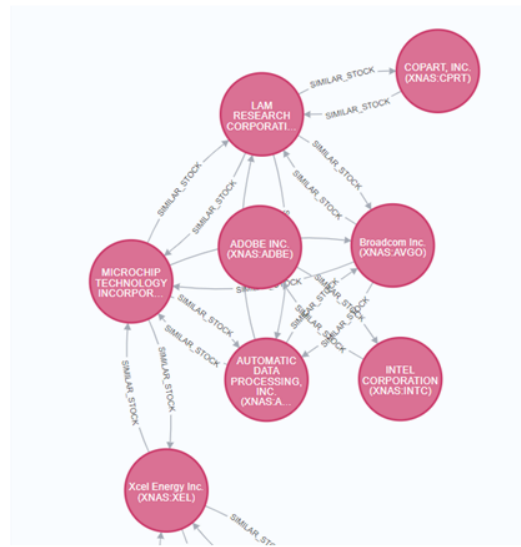


Group similar stock based on distance – stocks < 2.5 considered as SIMILAR



Apply graph algorithm such as Louvain to estimate clusters

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$



Diversification based on selecting stocks from different clusters, reducing exposure to highly central stocks etc. – minimize risk and optimize returns

# When Time Series Stock Data Meets neo4j (con.)

Algorithm	Description	Key Steps	Objective
<b>Louvain Community Detection</b> based on Price and Volume Correlation (KNN similarity algorithm)	Construct stock communities based on price and volume correlations to identify top performers.	<ol style="list-style-type: none"> <li>1. Compute KNN similarity score.</li> <li>2. Compute volume similarity score.</li> <li>3. Form communities using Louvain.</li> <li>4. Select top performers.</li> </ol>	Maximize returns and mitigate risks by selecting leading stocks from different communities.
<b>PageRank Algorithm</b>	Apply PageRank to identify influential stocks across the entire Nasdaq 100 using correlation-based stock communities.	<ol style="list-style-type: none"> <li>1. Calculate PageRank scores.</li> <li>2. Rank top stocks globally.</li> <li>3. Rank top stocks within communities.</li> </ol>	Identify and rank influential stocks to diversify across the broader market and within sectors to capture upside potential and also diversification
<b>Geodesic Distance</b>	Use Euclidean distance based on market cap, PE ratio, and industry to group similar stocks and apply Louvain clustering.	<ol style="list-style-type: none"> <li>1. Calculate Euclidean distance.</li> <li>2. Group similar stocks.</li> <li>3. Apply Louvain clustering.</li> </ol>	Optimize portfolio by selecting stocks from different clusters, reducing exposure to central stocks



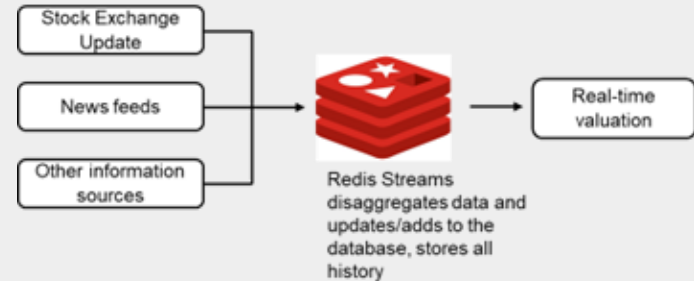


# redis for Advanced Stock Analytics

01

## Real-time Data Integration:

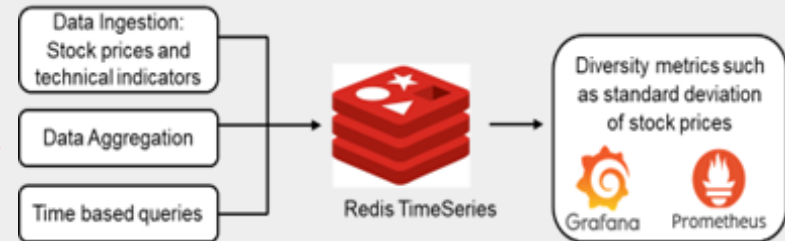
Existing database and data from other sources : stock market and news feed



02

## Diversity Metrics:

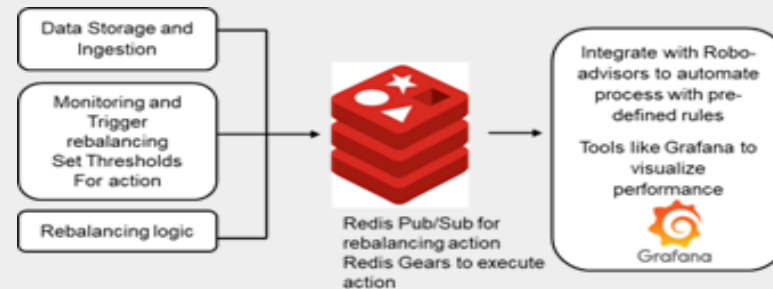
Sector allocation, geographical distribution, risk exposure - maintain well-balanced portfolio



03

## Automated Rebalancing:

maintain desired allocation over time adjusting to market movements & performance





# mongoDB for Advanced Stock Analytics

## Flexible Data Models

Stock data  
unstructured data

- Incorporate real-time stock data with news articles and social media sentiment
- Schema Flexibility makes it easy to add new data fields as needed
- Stores data persistently ensuring historical data is available for long-term analysis.
- Ideal for comprehensive data storage and querying

## Real-Time Data Processing

Market Trends  
Sentiment Analysis



- Collect real-time stock market data with high-performance write capabilities.
- Crucial for generating up-to-date market trends
- Store and analyze real-time social media feeds and news articles.
- Multiple "Point of View" which is suitable for integrating and analyzing diverse data sources

## Scalability

Horizontal Scalability  
Data Sharing

- Scale horizontally as data volume grows
- Distribute data across multiple servers to ensure fast processing
- Maintain rapid processing of analytical queries to provide timely insights to investors
- Supports complex queries and indexing, which are useful for detailed analysis and data retrieval




# When Redis & MongoDB meets Stock Analytics

	Application	Why RDBMS does not fit
 redis	Use as caching layer to store frequently accessed data such as stock price or pre-computed analysis. Real-time analytics due to in-memory storage and fast data access	Low latency due to ACID compliance, difficult to achieve real-time performance
 mongoDB	Ideal for data storage of diverse financial datasets - can handle complex queries, has flexible schema and horizontal scalability	Challenges with horizontal scalability and rigidity in schema makes it difficult to adapt to changing data structures

In the context of our business case of stock diversification, Mongo and Redis could also serve meaningful purpose:

- Data ingestion (historical data , financial reports) and pre-processing (cleaning, calculating basic metrics) and query and analysis- using Mongo
- Real-time processing (live tracking of stock prices and ongoing calculations) using Redis

# Conclusion /Summary

- To create a diversified portfolio with top performers, we identified a few ways that NoSQL databases can meaningfully add value.
- With  neo4j
  - KNN Similarity and Louvain algorithm could help identify clusters/community of closely correlated stocks based on price and volume, from which a portfolio can be formed with top performers from each community;
  - Page Rank could identify the most influential stocks globally or by community, which can be used to better describe the cluster, and choose stocks for portfolio formation;
  - Euclidean distance provides another way to quantify the similarity between stocks based on fundamental metrics such as market cap, PE Ratio, and industry, which can be used for investment decisions.
- With  redis
  - The unparalleled in-memory storage is a natural fit for real time stock data streaming and fast simple analytics, feeding data to Neo4j for fast analysis updates
- With  mongoDB
  - The flexible non-key search, together with the tolerance for many 'point of views' storage make MongoDB a great choice for complex pre and post-trade portfolio analytics

# Reference

- <https://neo4j.com/developer-blog/diversify-your-stock-portfolio-with-graph-analytics/>
- <https://medium.com/codex/pattern-driven-insights-visualize-stock-volume-similarity-with-neo4j-and-power-bi-13ca922acad1>
- <https://medium.com/graphed-trader/using-neo4j-for-graphed-trading-with-the-td-ameritrade-api-b6d1b46b779>
- <https://www.mongodb.com/resources/compare/mongodb-vs-redis>
- <https://www.kaggle.com/datasets/svaningelgem/nasdaq-daily-stock-prices>
- <https://neo4j.com/docs/browser-manual/current/visual-tour/>
- <https://redis.io/docs/latest/operate/rs/references/metrics/>
- <https://redis.io/docs/latest/develop/>