Optimizing Recommender Systems and Warehouse Organization Through Graph Database Utilization

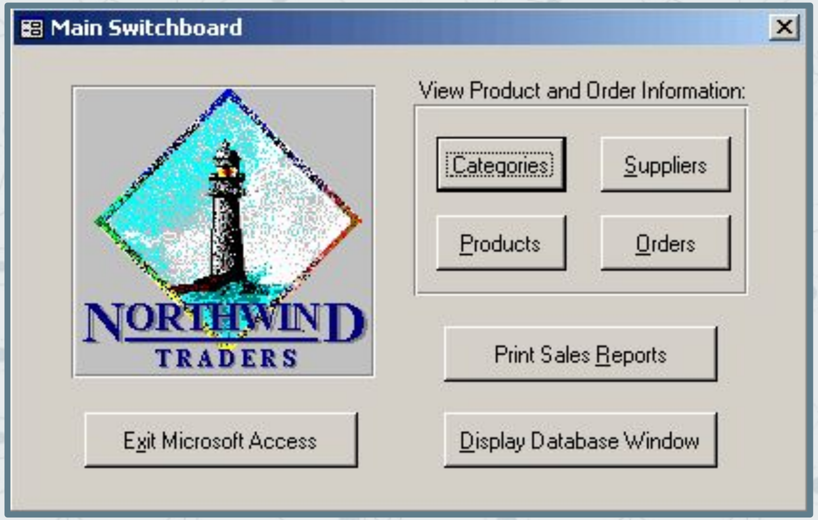
UC Berkeley MIDS, Summer 2023

Nathan Arias

Nat Bussion

Divya Menghani

Kirti Prakash

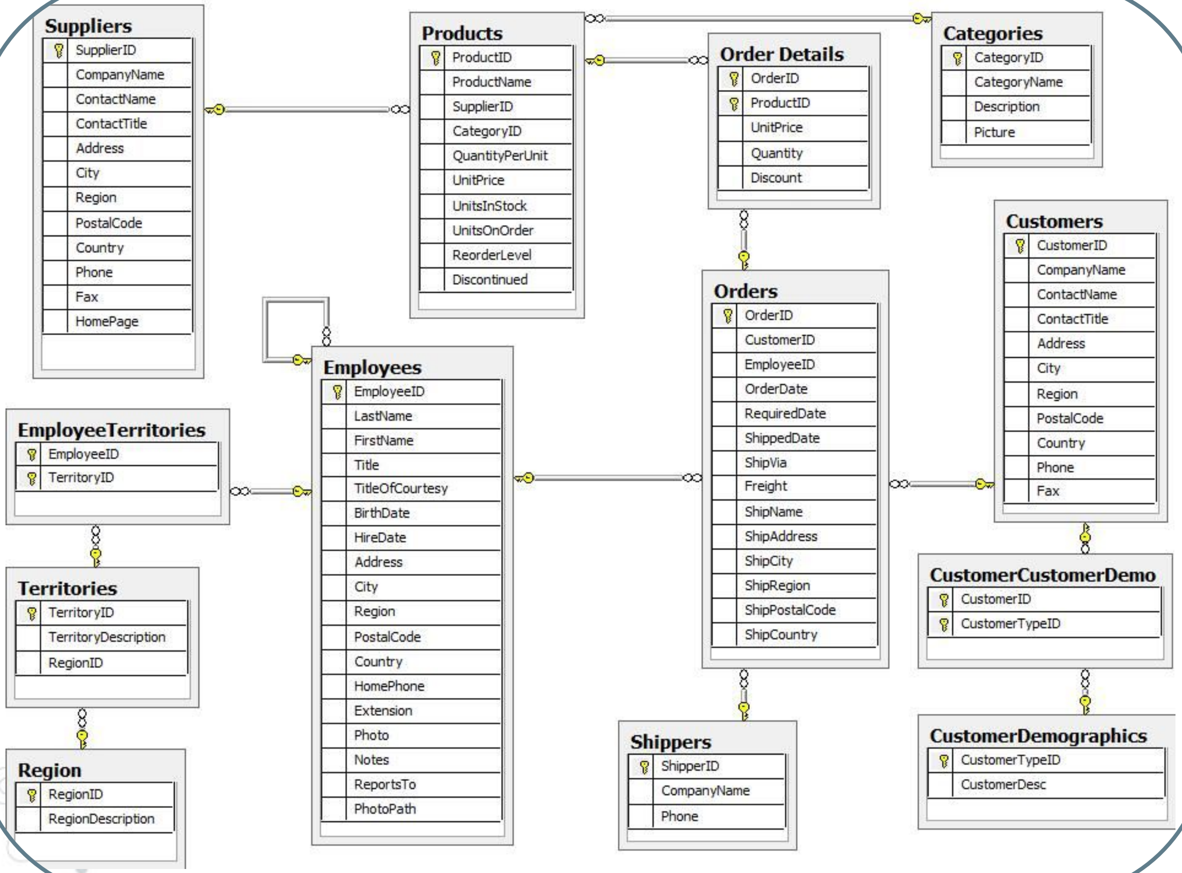


The Northwind Database

Northwind Traders sells specialty food products sold online from around the world.

The Northwind database contains purchase history data for Northwind Traders including suppliers, products, orders, customers and more.

The Northwind ERD



Business Demand: Find products frequently purchased together

neo4j

Leveraging Graph Databases

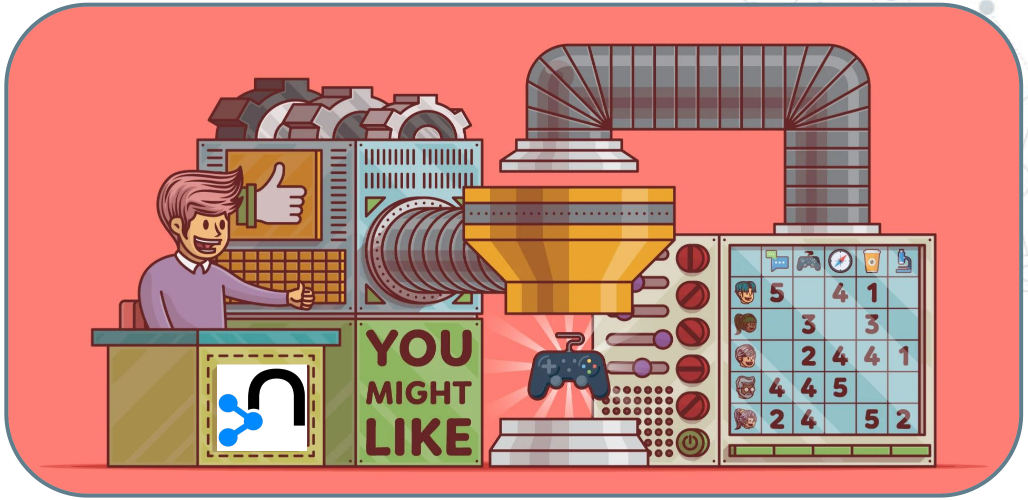
Business Use Cases

1) Product recommender system

2) Warehouse optimization

Business Use Case 1:

Product Recommender system

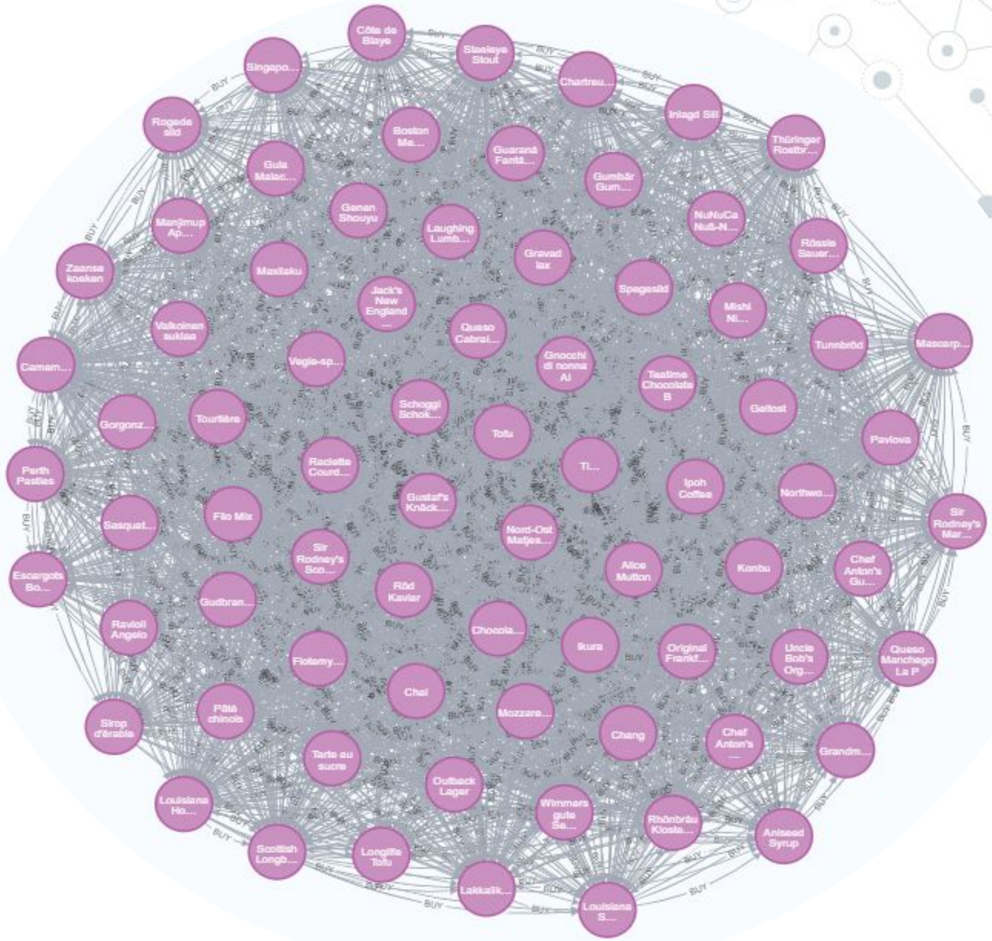


Using graph database does not immediately grant useful insights into the relationships between products.

However underlying structure can be revealed through careful use of graph algorithms

Our Recommendation System will use the following algorithms:

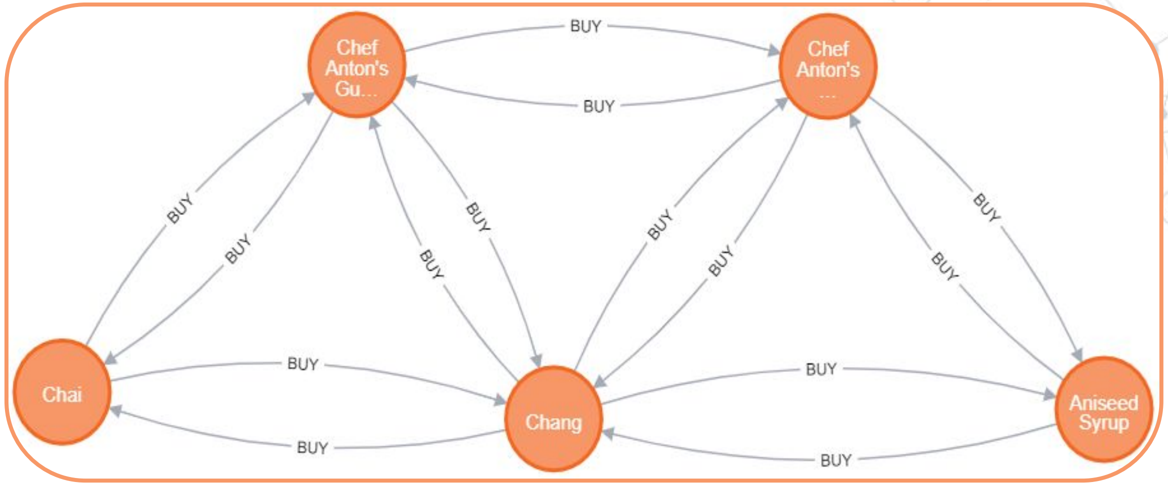
1) Triangle Counts
2) Local Clustering Coefficient
3) Community Detection



Triangle Count Algorithm

Triangle Count Algorithm:

- Detects communities in graphs.
- Used to determine the number of triangles passing through each node in the graph
- A triangle is three interconnected nodes



Triangle Count in Recommender Systems:

- Reveals underlying community structures.
- Enhances recommendation accuracy by leveraging node relationships.
- Identifies closely-knit groups, aiding in targeted suggestions.

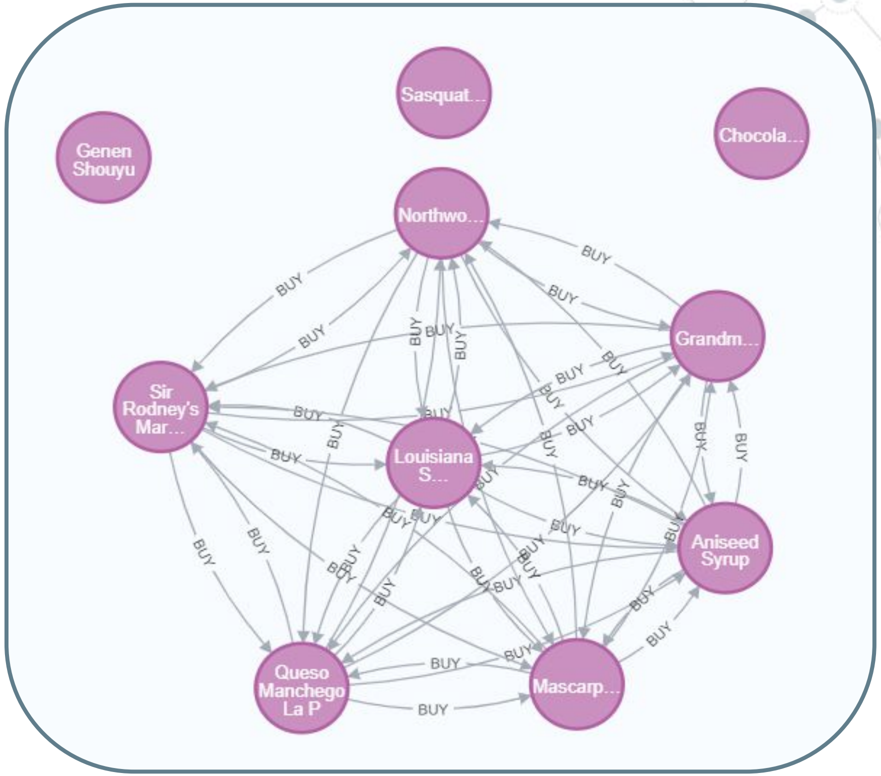
Local Clustering Coefficient Algorithm

Local Clustering Coefficient Algorithm:

- Describes the likelihood that two neighbors of a given node are neighbors themselves
- Output is a fraction that takes on a value between 0 and 1.

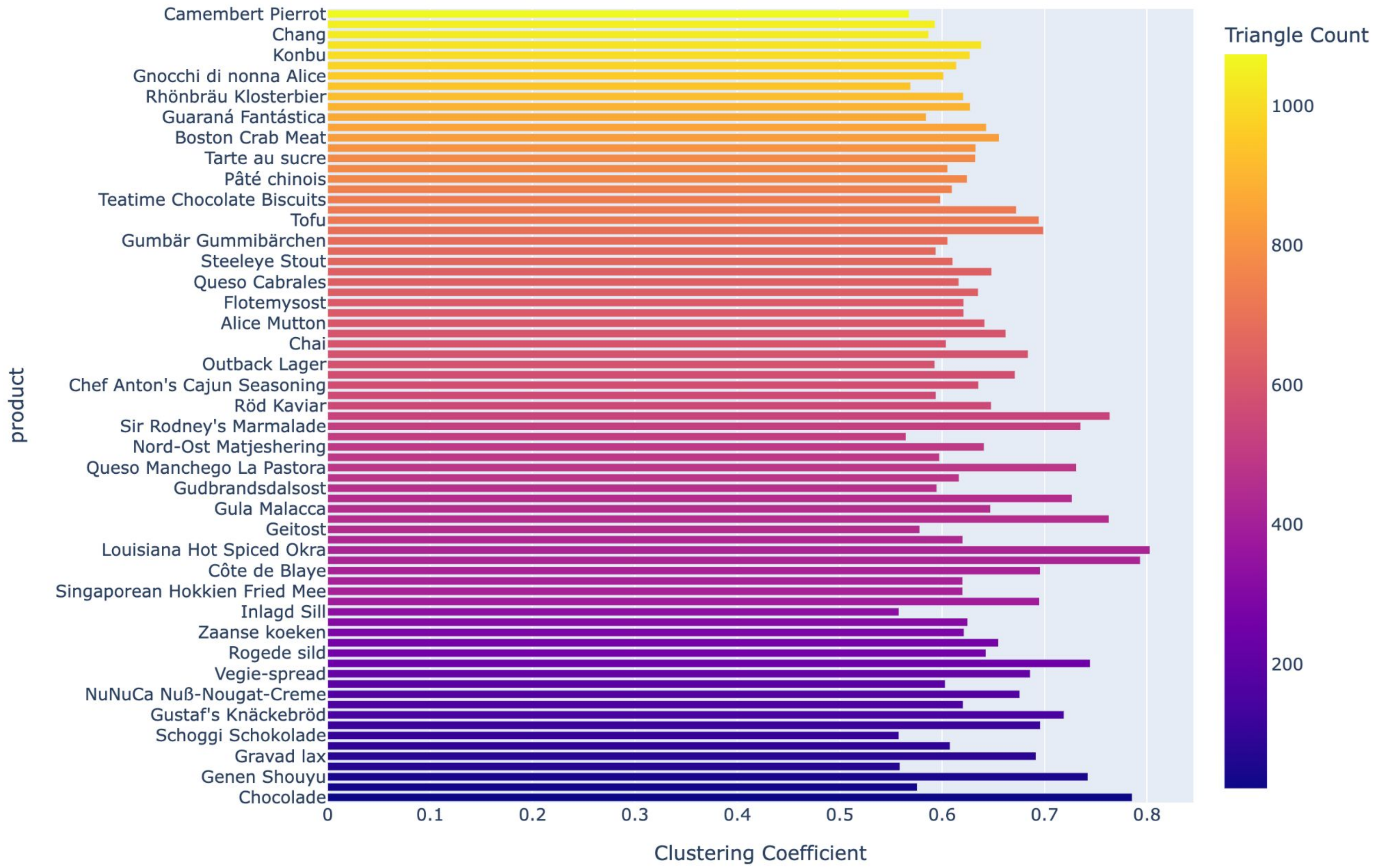
Clustering Coefficient in Recommender Systems

- Measures the interconnectedness of a product's immediate network.
- Indicates how likely two products connected to a given product are to be connected to each other.

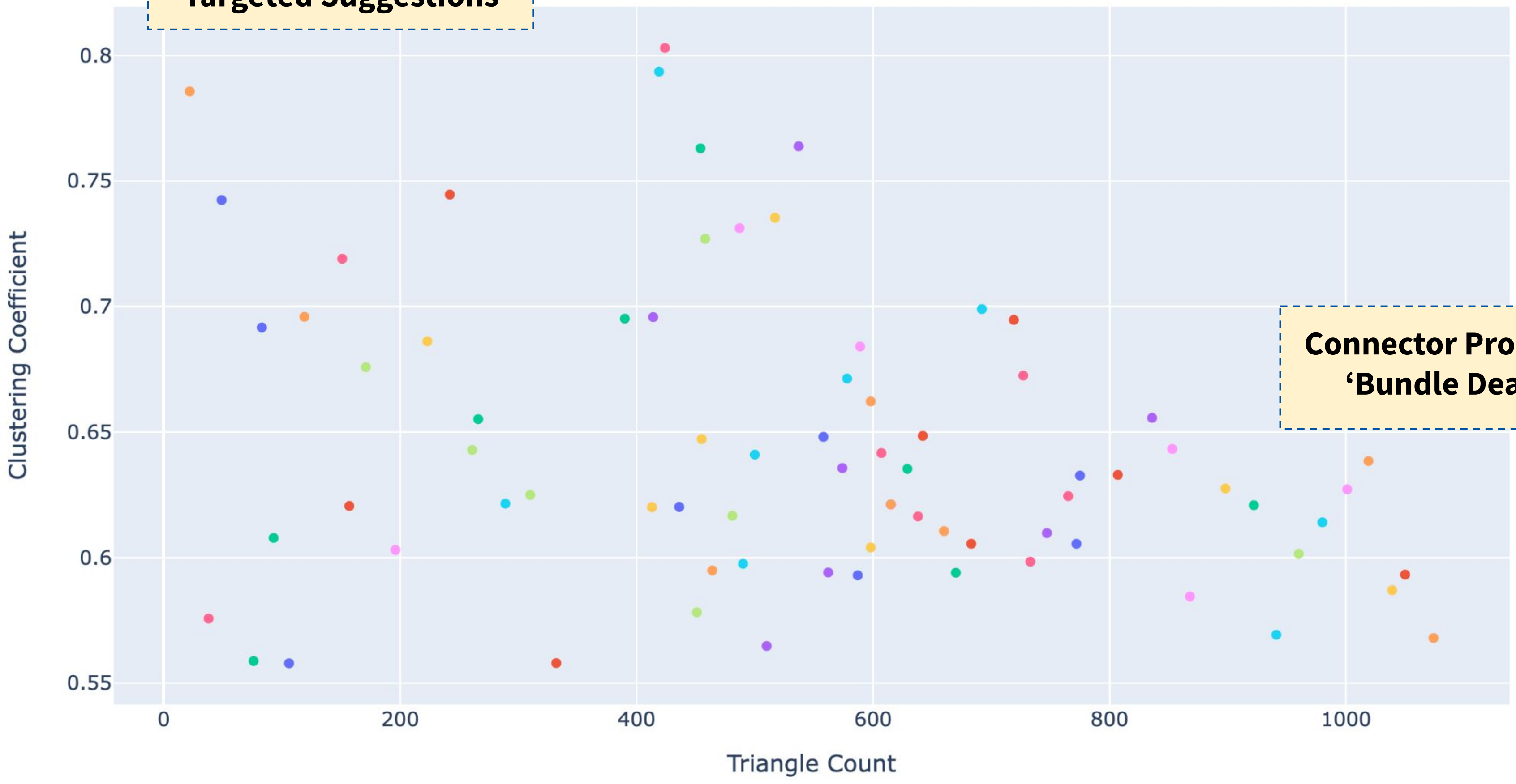


10 products with highest clustering coefficient

A Holistic View of Triangle Counts and Local Cluster Coefficients

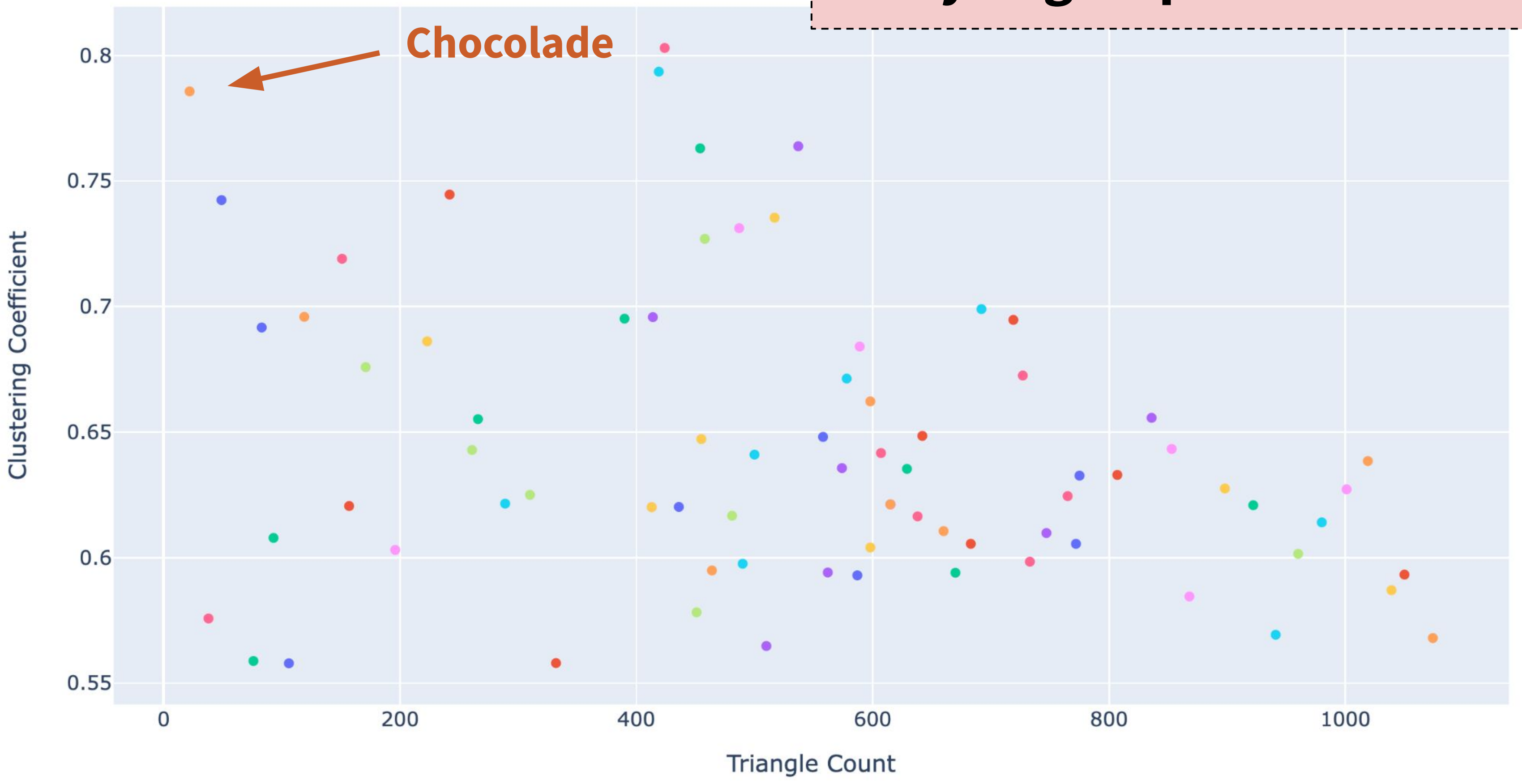


Tight Knit Communities
'Targeted Suggestions'



Connector Products
'Bundle Deals'

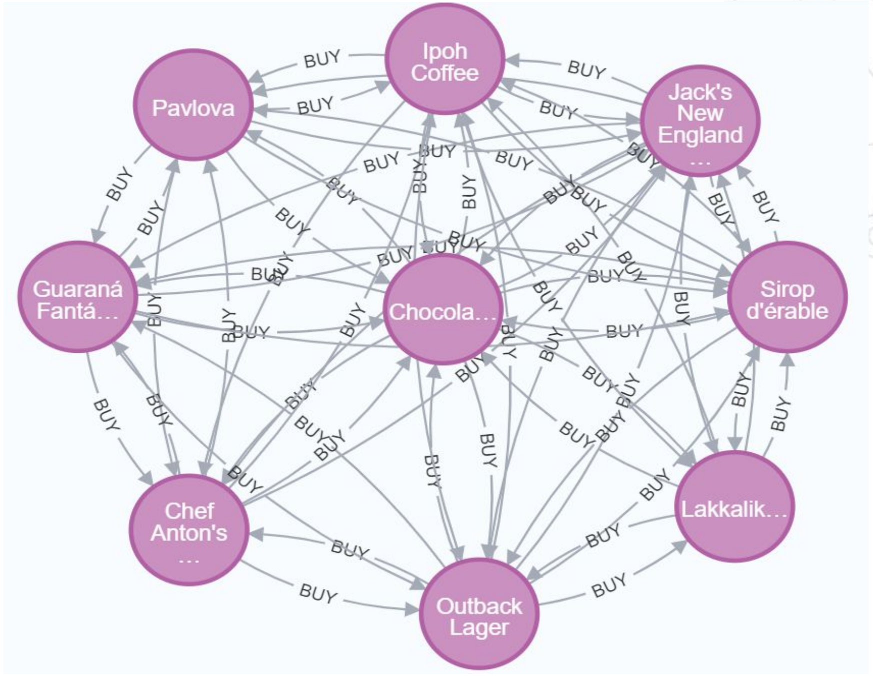
Analyzing a Specific Product



Product Example - Chocolade

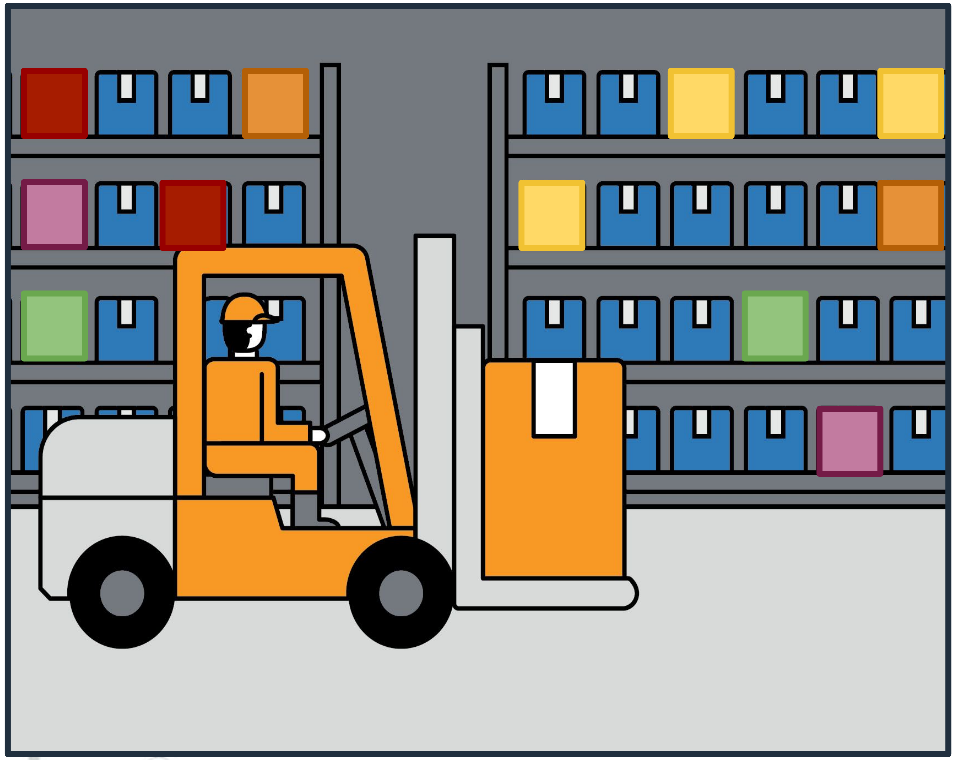
In this graph we can see that a central node "Chocolade" which has a triangle count of 22, clustering coefficient of 0.78 and is closely related to 8 other products.

A small triangle count and high clustering coefficient relative to other products indicates that this product exists in a niche of specialty products which are shown here.



product	Clustering Coefficient	Triangle Count
Chocolade	0.785714	22

Business Use Case 2: Warehouse optimization



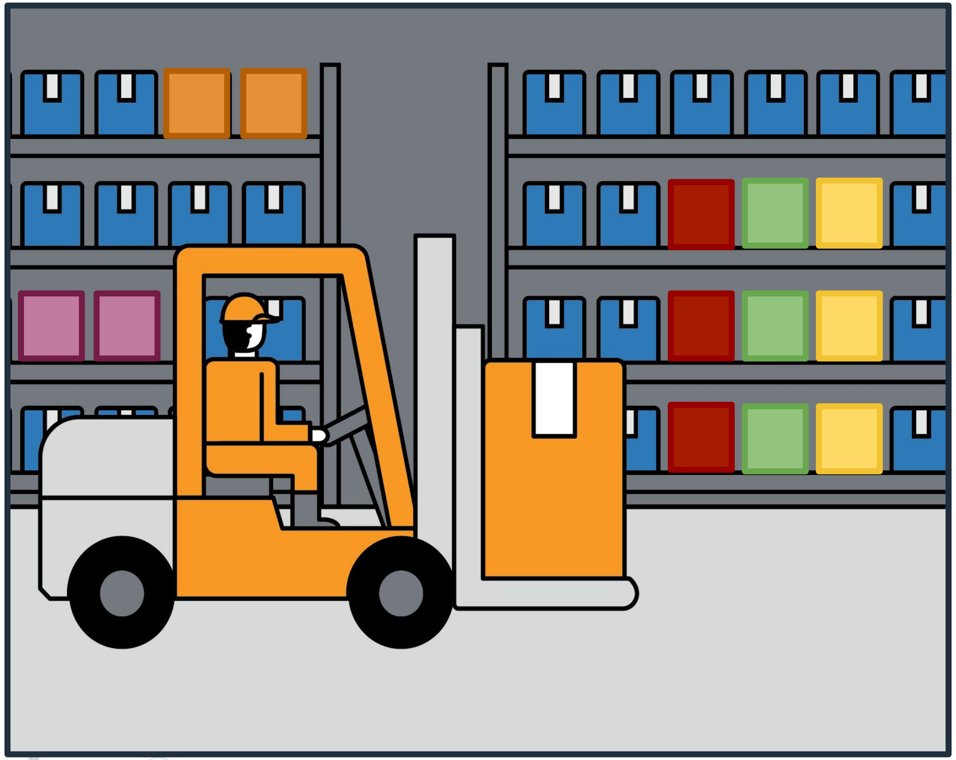
Example Order

🟥 x 1 🟨 x 1

🟩 x 1 🟪 x 1

🟧 x 1

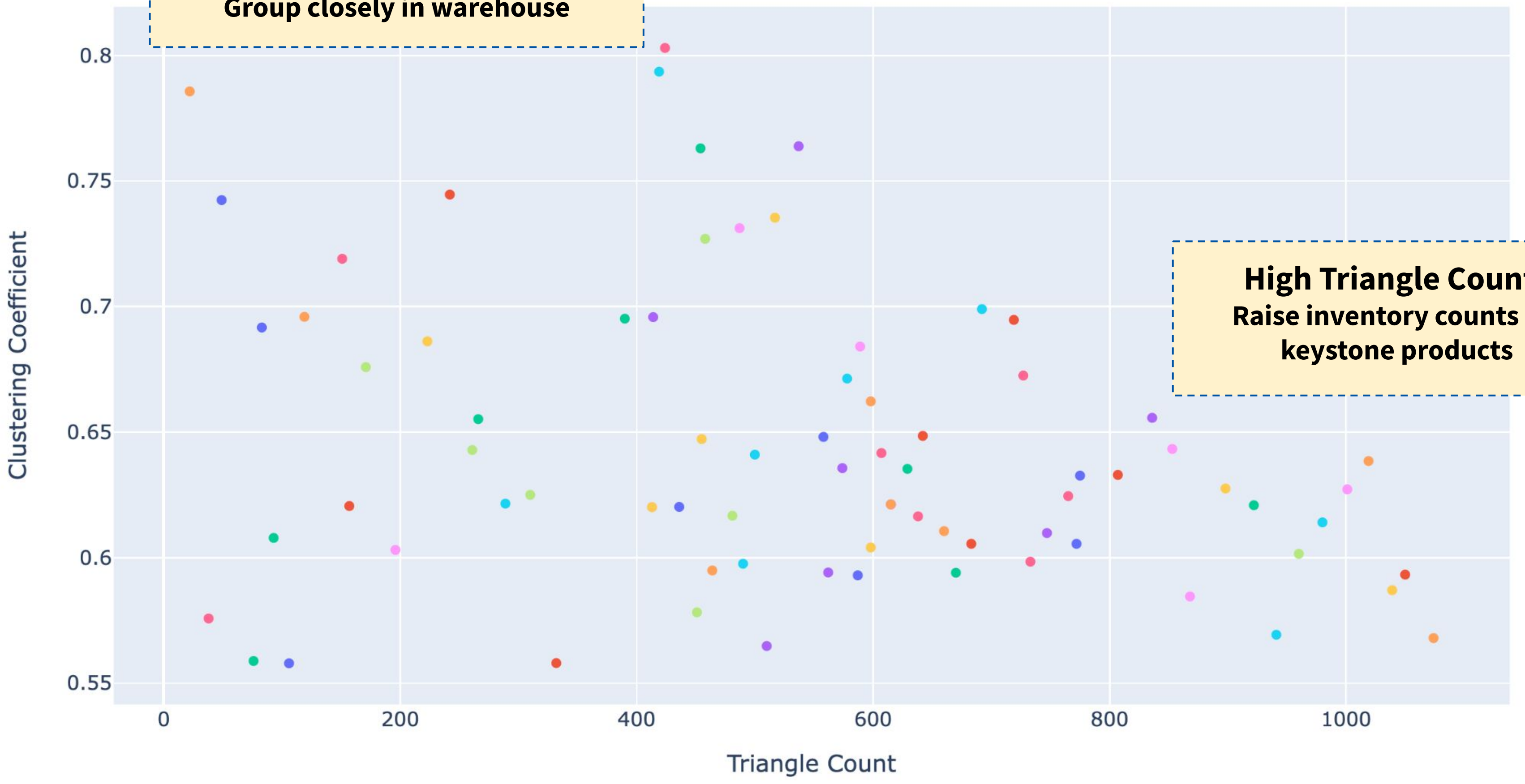
A **representative** order exhibits the difficulties experienced by warehouse order specialists. Products often are located in different tiers or bins requiring additional time investment.



Products Frequently Sold Together 🟥 🟩 🟨

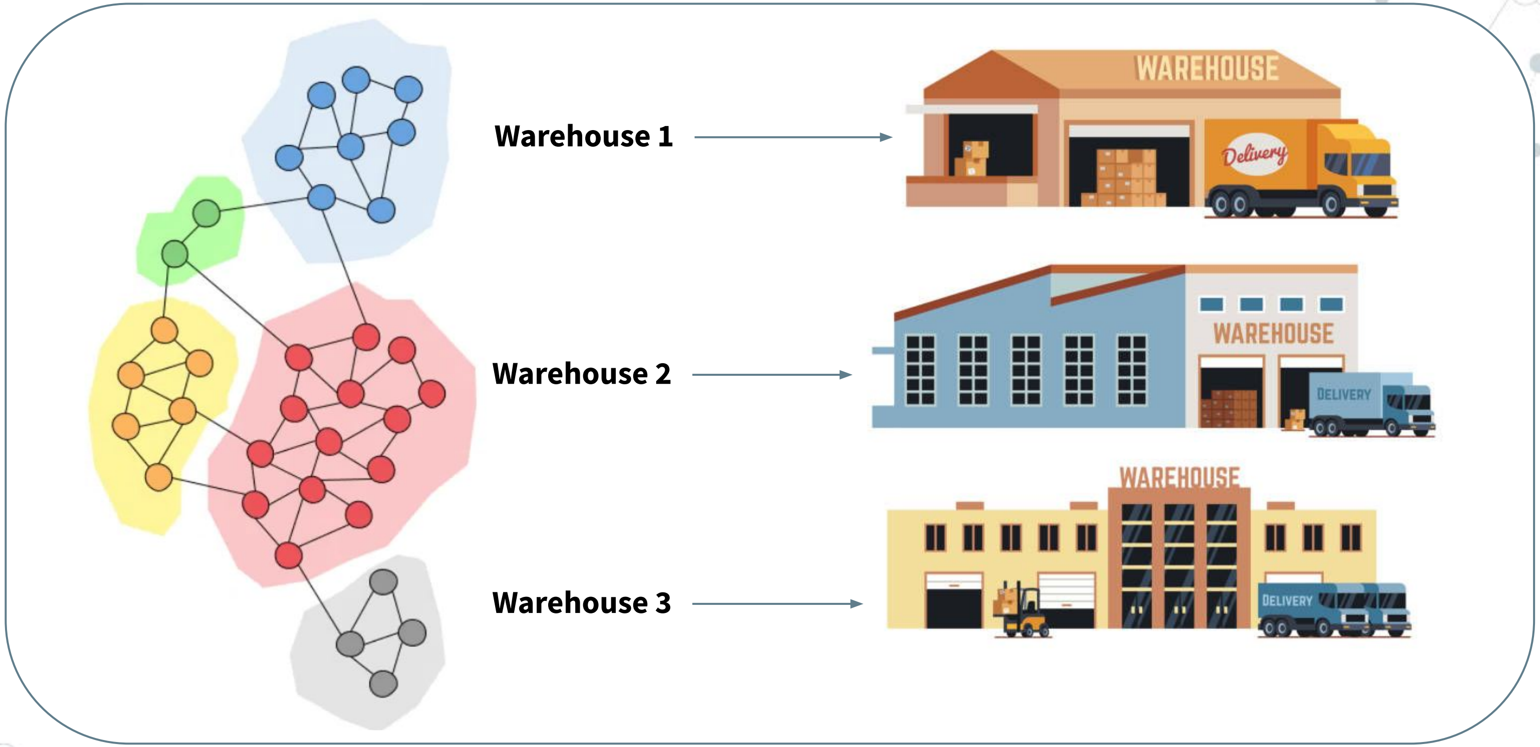
Improved understanding of product sales and **interproduct relationships** can make tangible improvements on warehouse order generation.

High Clustering
Group closely in warehouse



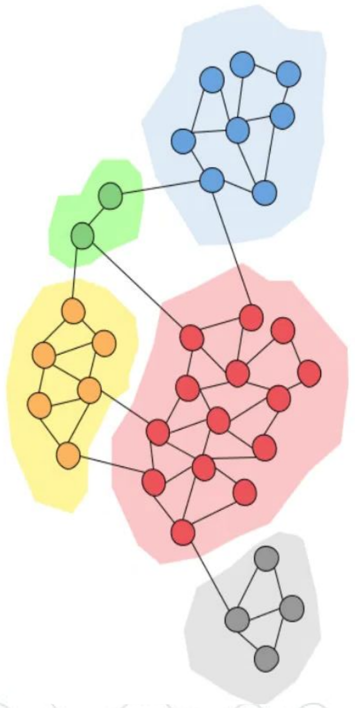
High Triangle Counts
Raise inventory counts for keystone products

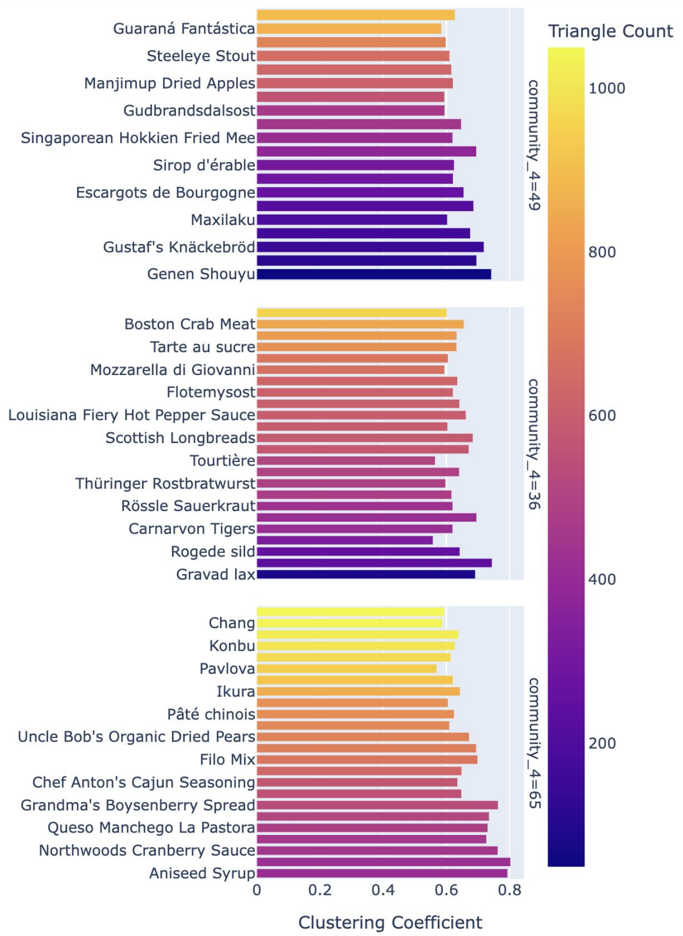
Louvain Modularity



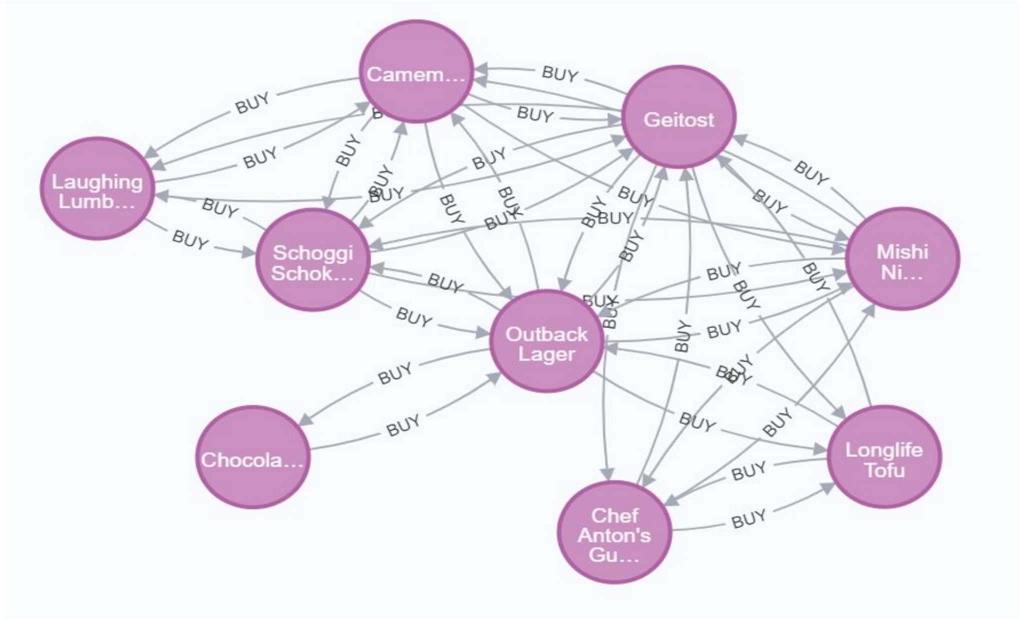
Louvain modularity is a method used to identify clusters or communities in networks by evaluating and optimizing the connections between nodes.

Louvain Modularity - Minimum Community Size 15



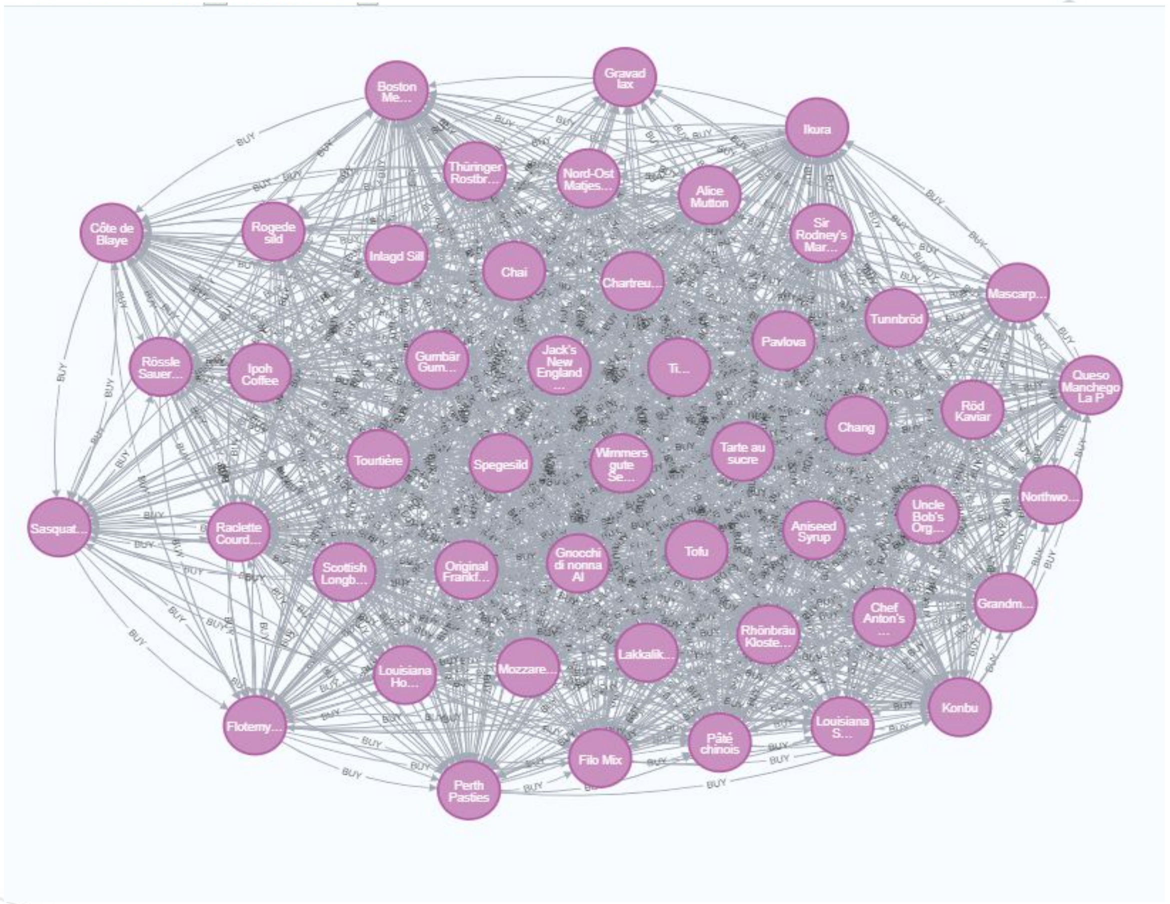


Louvain Modularity Graph For Community id 4



Louvain modularity algorithm it tends to assign a community id to all the products and keep similar products under the same community.

Louvain warehouse community size 24 - 48 nodes have a community size = 24



Exploring MongoDB

Solutions with MongoDB:
MongoDB excels at handling vast volumes of high velocity, diverse data. We can use MongoDB to store multiple points of view (POVs) for the data generated by this system for analytical purposes.

POVs that would be of interest to us include: Highest Product Triangle Count, Highest Clustering coefficient, Communities views to allow fast queries when giving recommendations while customer browses or ordering a product.

Additionally, we could use MongoDB for web server and/or web API to avoid querying and joining multiple relational tables while users navigate our product catalog.

Why not a Relational Database?
Traditional relational databases can struggle with schema changes and may not scale horizontally as efficiently. This makes them less suited for analytics where data is needed at high velocities.

Exploring Redis

Solutions with Redis:
Redis, as an **in-memory** data structure store, is ideal for caching frequently accessed data. Product details and ‘lightning deals’ can be stored in Redis, ensuring fast access and reduced strain on primary databases.

We can store inventory data in real time and do quick global update of inventory levels. We ensure a product one customer is buying at cannot be sold to another.

Why not a Relational Database?
Relational databases would need to fetch data from disk, which is slower than fetching from memory. During high-traffic events like flash sales, this can lead to significant lag or even system failures, compromising user experience and potentially leading to lost sales.

Questions?