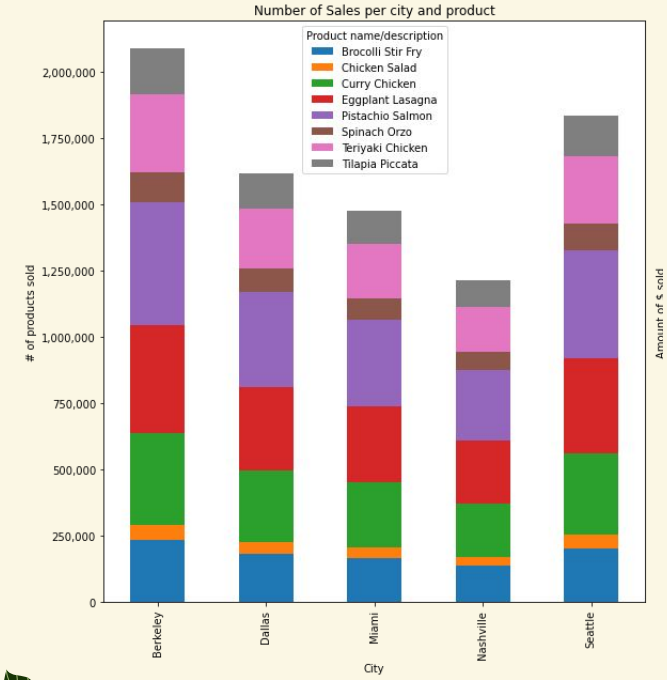
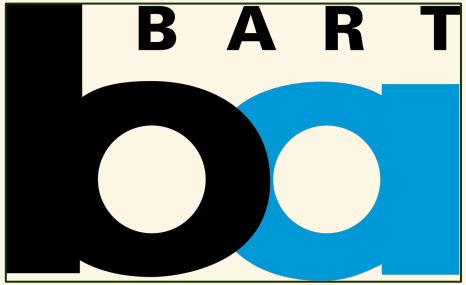
ACME Gourmet Meals:
Using NoSQL to achieve our vision

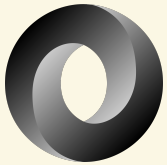
Mauricio, Thomas, Sailesh, Mei
DATASCI 205

AGM's Vision


Number of Sales per city and product
Product name/description
Brocolli Stir Fry
Chicken Salad
Curry Chicken
Eggplant Lasagna
Pistachio Salmon
Spinach Orzo
Teriyaki Chicken
Tilapia Piccata
# of products sold
2,000,000
1,750,000
1,500,000
1,250,000
1,000,000
750,000
500,000
250,000
0
Berkeley
Dallas
Miami
Nashville
Seattle
City


BART

AGM's Data-science Tech



Stores

Products

Customers

Line Items

Sales

Holidays

States

Zip_codes

Cities

- Schema Flexibility
- Scalability
- Performance (Pending the use case)
- Ease of Handling Unstructured or differently structured data

- Graph relationships
- Optimized for Pathfinding
- Data visualization

- In-Memory Speed
- Real-time analytics
- Pub/Sub Messaging

- Dynamic Data Storage
- Non-Complex Joins
- Rich Query language

Neo4j: Using BART to deliver meals

AGM is considering partnering up with BART, and we can leverage Neo4j to construct a graph of the BART network and optimize for travel distance from our stores.

Constructing the graph:

1. Customers, stores, and stations will be represented as nodes
2. Connections between stations and from stations to customers will be represented by edges

Depending on our goals, we can modify our edge weights to optimize for any objective.

a. Optimizing for faster deliveries can be achieved by having edges be travel times and combining that information with departure times
b. Optimizing for cost can be achieved by having additional edges between stations representing BART fares

Dijkstra's: Shortest travel time using BART

Construct a graph with:

1. Nodes for stores in the area
2. Nodes for each customer
3. Edges are travel times between stations

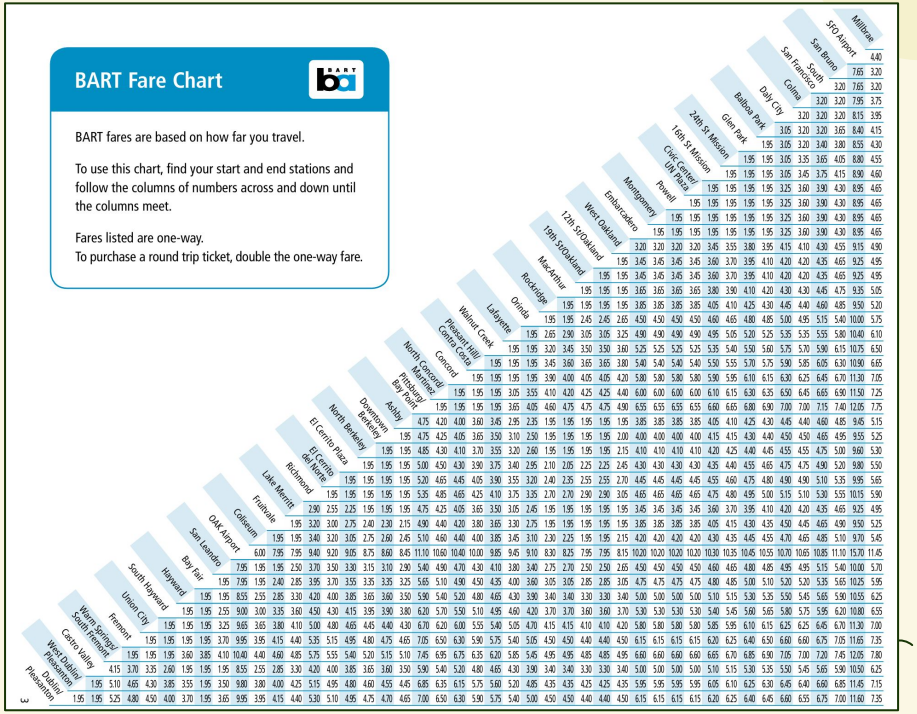
Using Dijkstra's, we can find the distance from each customer to their closest BART, then from their closest BART to the closest BART next to an AGM location. If they live further from the AGM location than the closest BART station, then BART could be an efficient delivery option.

Example: Cost for delivering using BART

Construct a graph with:

1. Nodes for stores in the area
2. Nodes for each customer
3. Edges are BART fares. Fares are publicly available on the BART website

We can incorporate costs for traveling using BART in situations where we would need to make multiple stops and leaving at multiple stations during the trip.

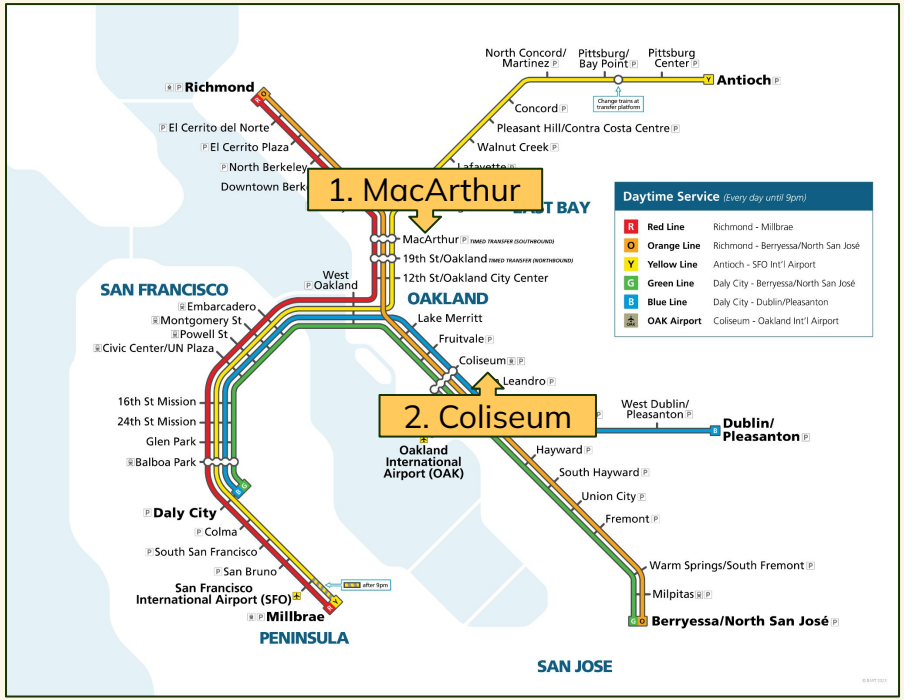


Betweenness: Finding the Most Impactful Stations

Construct a graph with:

1. Nodes for stores in the area
2. Edges are travel times between stations

Using Betweenness, we can identify the key transit stations that act as crucial intermediaries in the network. These nodes are essential for efficient movement and connectivity, as they lie on the shortest paths between other locations.

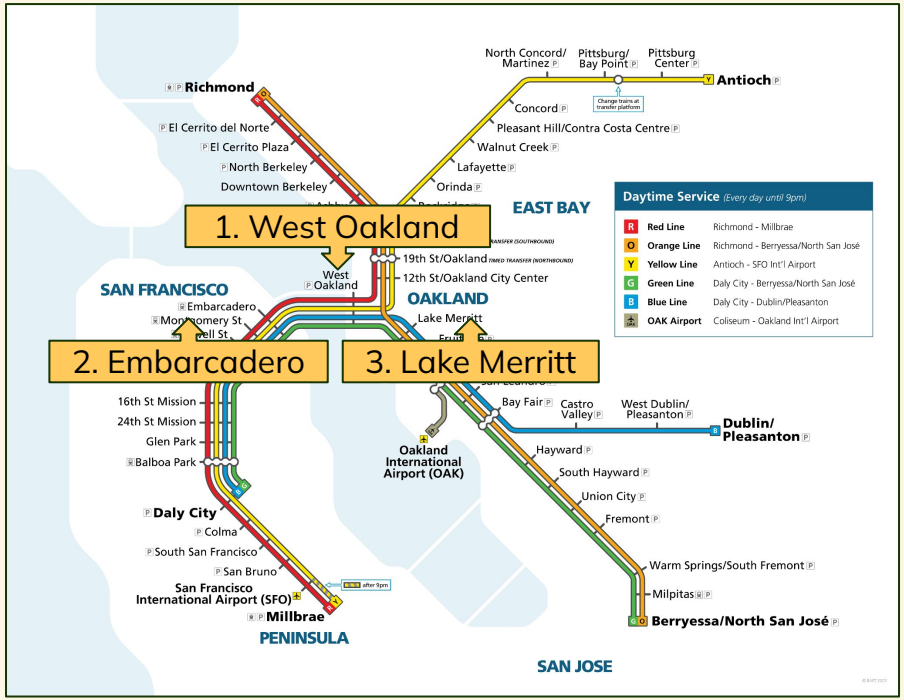


Centrality: Finding Optimal New Store Locations

Construct a graph with:

1. Nodes for stores in the area
2. Edges are travel times between stations

Using Harmonic Centrality, we can identify the most accessible transit stations from all other locations in the network. We can select future store locations near well-connected transit hubs, ensuring ease of access for customers and optimizing foot traffic.

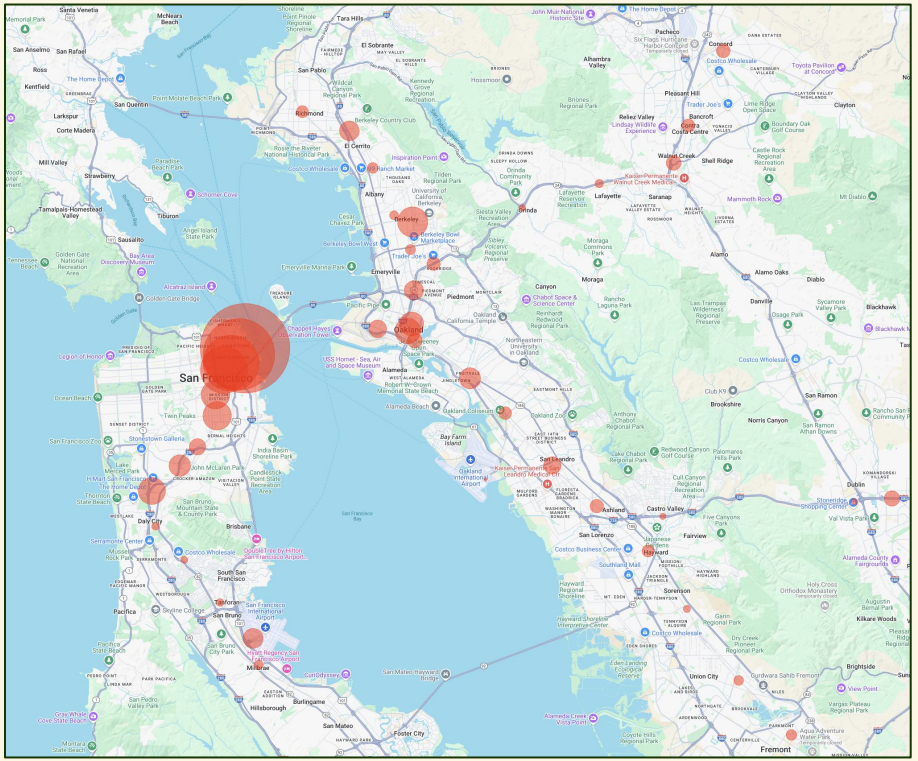


Neo4j: Optimizing locations by communities

When considering optimal pickup or distribution centers, we can cluster stations into communities. We can leverage Bart's ridership data for clustering to optimize stations with many riders.

First, we need to create a ridership table for each station. This is done in 3 steps:

- Import Bart's station data from Excel workbooks
- Map 2 letter station code to station name
- Normalize station exit counts

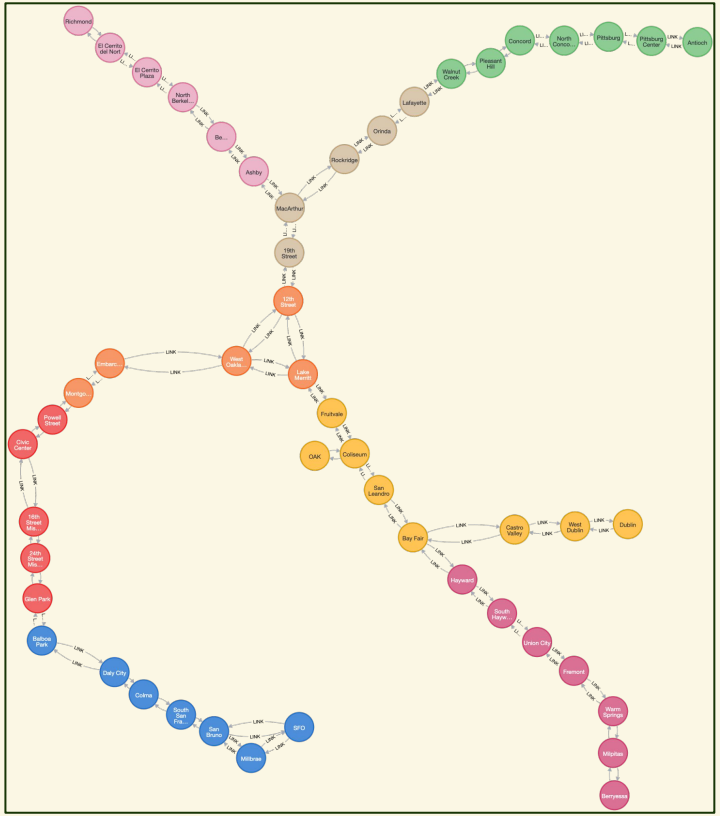


Neo4j: Optimizing locations by communities

Using Neo4Js Louvain Modularity algorithm, we can cluster stations into 8 communities.

Clustering can be tuned as needed based on:

- Ridership
- Population
- Travel time



MongoDB: Enhancing product descriptions

Products can have highly varying product descriptions. MongoDB elegantly handles new attributes without disrupting the database schema.

1. Seasonal meals or promotional items (e.g. holiday specials, celebrations) would have a sparse encoding in relational databases but are efficiently stored by MongoDB.
2. We would gain the ability to efficiently store meals with diverse ingredients and dietary tags, such as gluten free, vegetarian, halal, and more, while also allowing for them to be quickly searched and filtered through by customers.

Redis: Potential drone delivery

1. Fast in-memory performance (real time updates and queries - SQL might lag when dealing with rapid updates)
2. Pub/Sub messaging available to monitor any updates in real-time (SQL would need an extra layer to accomplish this in real-time)
3. Key-Value data prone (e.g. drone: DRN01) allowing faster look-up and updates (SQL will require querying tables and might be slower)

Key advantages of Redis over SQL:

- Real-time updates (no risk of disk-based storage)
- Capacity for live notifications built into the code
- Optimized for high frequent writes and edits

Thank you!