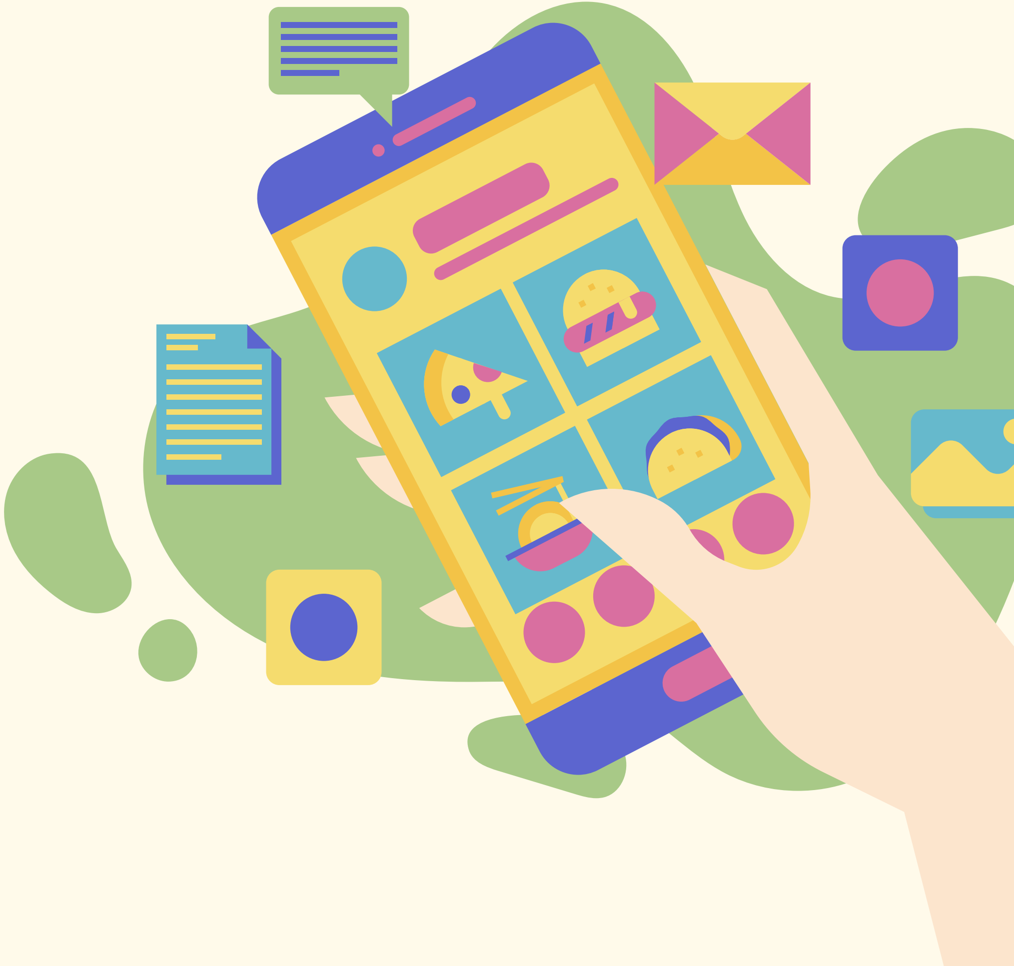
AGM Food Delivery using BART

Robert Meyer,
Brendan Foo,
Angikaar Singh Chana



Overview

01

Project Introduction

Overview of our project's problem statement and solution approach

02

Louvain & PageRank

Neo4j algorithms to designate clusters and cluster centers

03

Delivery Path Optimization

Neo4j algorithms to develop efficient delivery paths

04

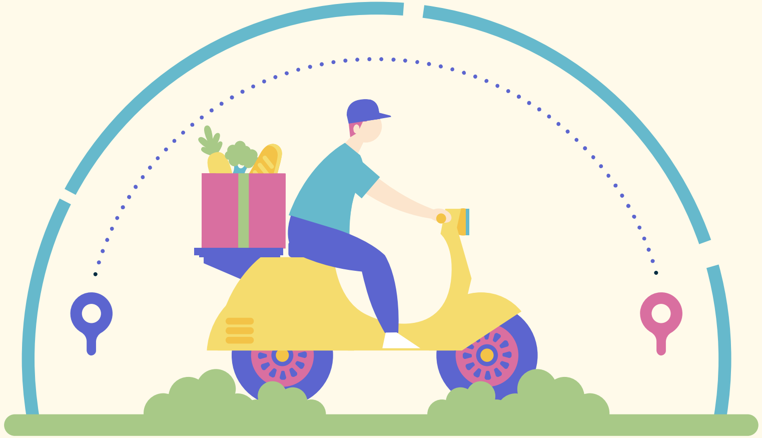
Redis & MongoDB

Possible further implementations of other graphical databases

Project Introduction

- Our company, AGM, is interested in expanding its food delivery network across the Bay Area
- For economic and environmental reasons, AGM is interested in leveraging public transit for its food deliveries, such as BART
- Our approach was to:
  - create a Neo4j graph to relate the BART stations as nodes,
  - designate sections of the Bay Area as clusters, finding BART stations that can act as cluster centers
  - develop an algorithm to determine the shortest path to said cluster centers, and,
  - recommend the installation of AGM delivery pickup centers at these BART stations
- We use a graphical database, as our data is not tabular and a relational database would not be as efficient

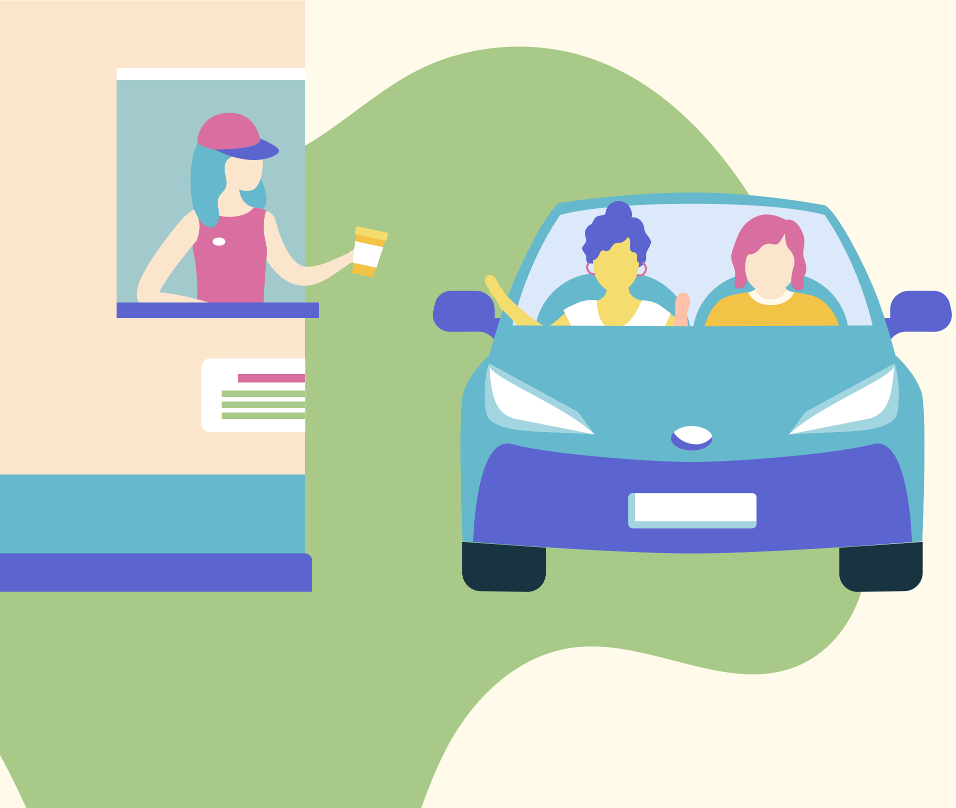
Food Delivery Model



Our implementation of the project falls under the following food delivery model:

1. A robot/employee moves the food from our AGM Berkeley location to the closest BART station, which we assume is Downtown Berkeley
2. The food is loaded into an AGM-specific BART car on the various different BART lines with the supervision of an AGM robot
3. The food travels to different hubs, where the AGM robot loads the food into AGM food storage lockers
4. The food recipient, whether it be a customer or a partnering delivery service (e.g. DoorDash), picks up the food and the transaction is complete for AGM

➢ The benefits of this model are reduced service fees and labor costs, since BART streamlines the delivery process



Louvain

Determining the optimal number of communities (clusters)

Louvain Clustering Algorithm

- Louvain algorithm assigns each node as a community and then collapses nodes into nearby communities, maximizing the intra-community edges while minimizing the inter-community edges
- This algorithm yielded that the optimal number of communities is 11, suggesting that we should install 11 hubs – where to install is answered by a centrality algorithm

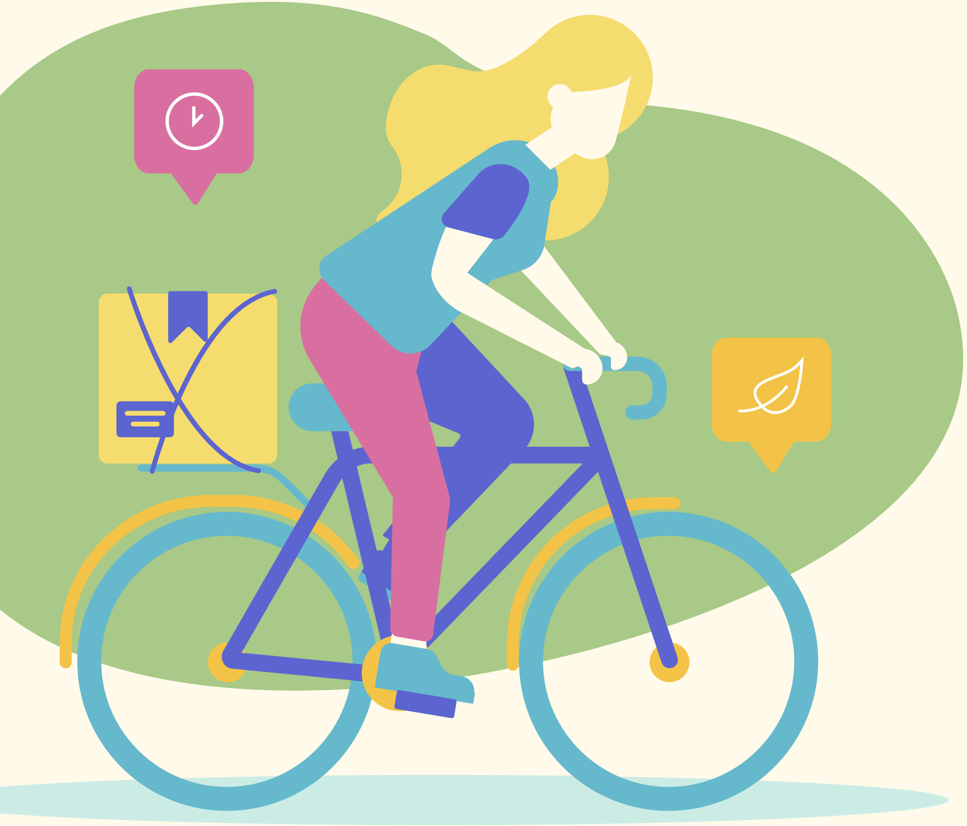
```
query = """

CALL gds.louvain.stream('ds_graph', {includeIntermediateCommunities: true})
YIELD nodeId, communityId, intermediateCommunityIds
RETURN gds.util.asNode(nodeId).name AS name, communityId as community, inte
ORDER BY community, name ASC

"""
```

```
len(df['community'].unique())
```

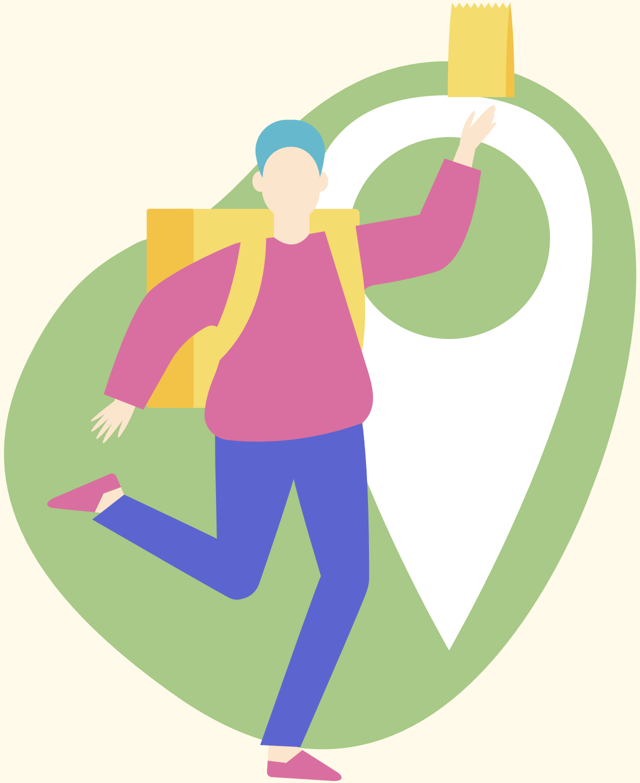
```
11
```



PageRank

Determining central delivery hubs

Ranking Communities for Delivery



We used a PageRank algorithm

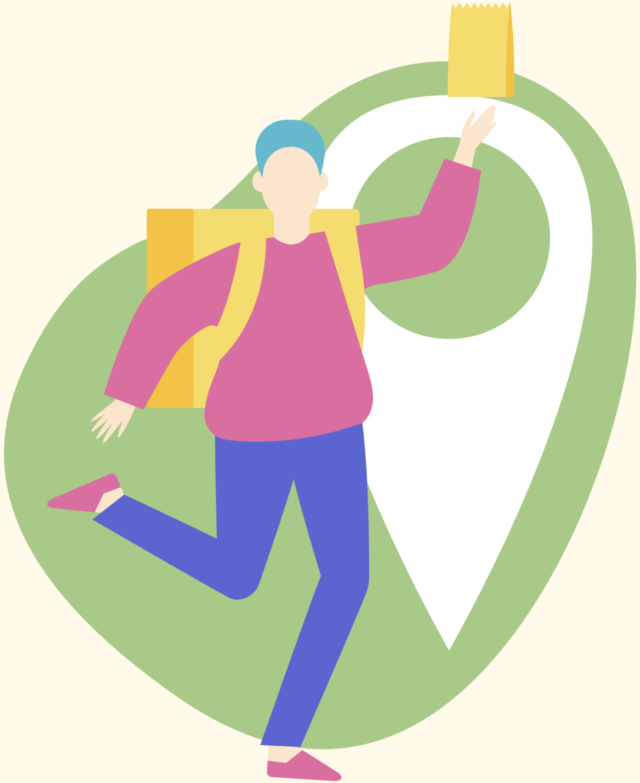
1. Determined PageRank score for each BART station based on connectivity and accessibility

2. Referenced the grouped BART stations by communities from Louvain Clustering Algorithm

3. Sorted BART stations by PageRank scores to determine most central station

This will help streamline the delivery process by providing a centralized location

Ranking Communities for Delivery

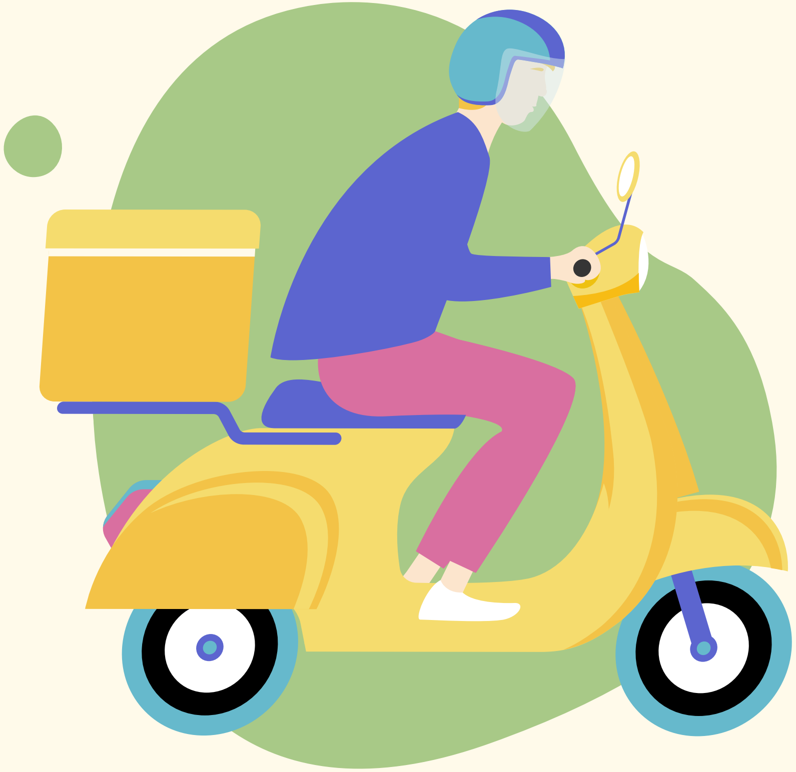


Most Central Delivery Hubs

0	MacArthur
1	24th Street Mission
2	Balboa Park
3	Milpitas
4	Bay Fair
5	Coliseum
6	Montgomery Street
7	El Cerrito del Norte
8	San Bruno
9	Pittsburg Center
10	West Oakland

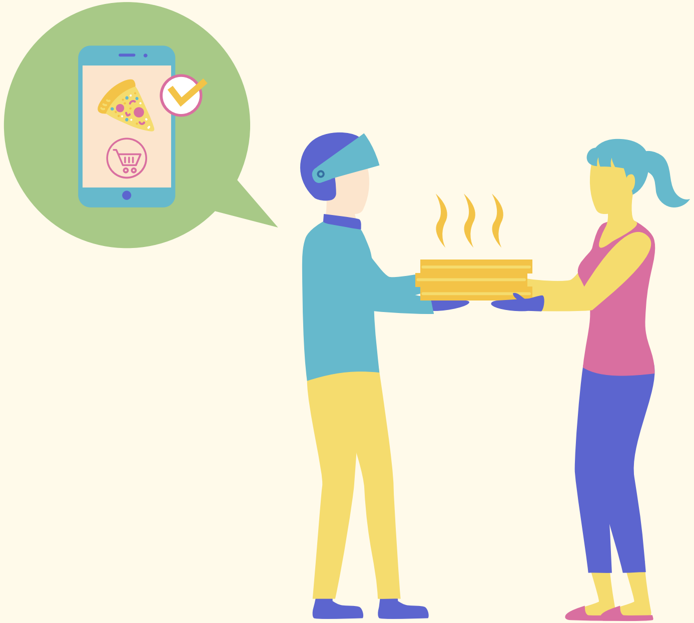
Shortest Path

Determine most efficient routes for food delivery



Shortest Path Algorithm

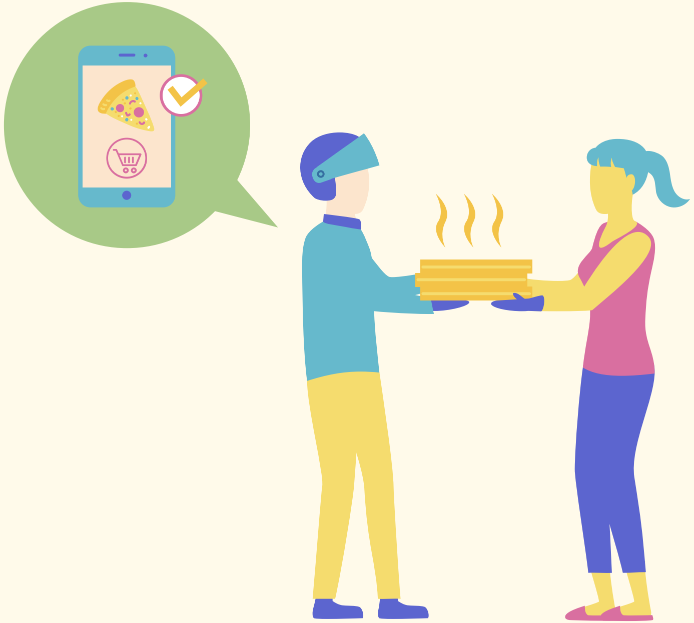
- Determine the most efficient Bart routes to get from our kitchen station to one of the delivery hubs
- Help delivery companies optimize their delivery routes and minimize the time and distance required to transport food
- We find the least number of hops and we use Dijkstra's algorithm to find the path between nodes with the lowest cost.
- By combining our pagerank algorithm and shortest path, we determine the shortest and most efficient routes between the most important and accessible hubs.



Shortest Path Algorithm

From Downtown Berkeley to Pittsburg Center

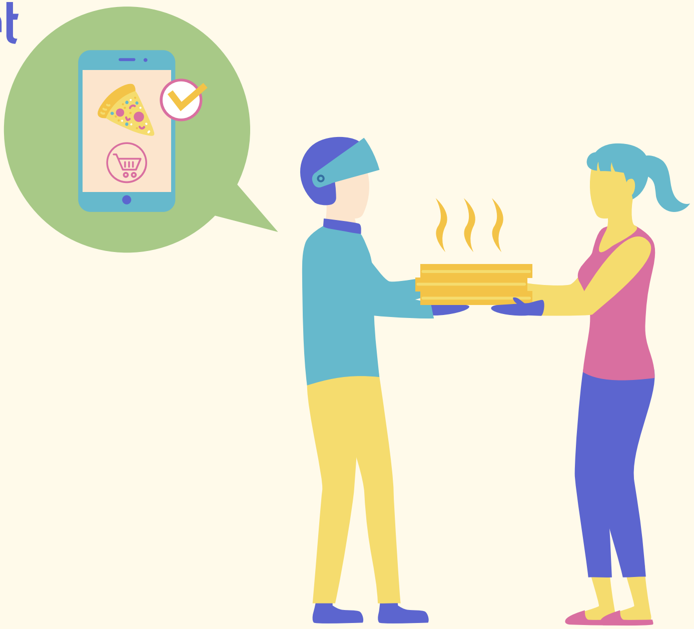
```
--------------------------------
   Total Cost:   3239
   Minutes:   54.0
--------------------------------
depart Downtown Berkeley, 0, 0
orange Downtown Berkeley, 0, 0
orange Ashby, 180, 180
orange MacArthur, 240, 420
yellow MacArthur, 59, 479
yellow Rockridge, 240, 719
yellow Orinda, 300, 1019
yellow Lafayette, 300, 1319
yellow Walnut Creek, 300, 1619
yellow Pleasant Hill, 120, 1739
yellow Concord, 360, 2099
yellow North Concord, 180, 2279
yellow Pittsburg, 360, 2639
yellow Pittsburg Center, 600, 3239
arrive Pittsburg Center, 0, 3239
```



Shortest Path Algorithm

From Downtown Berkeley to Montgomery Street

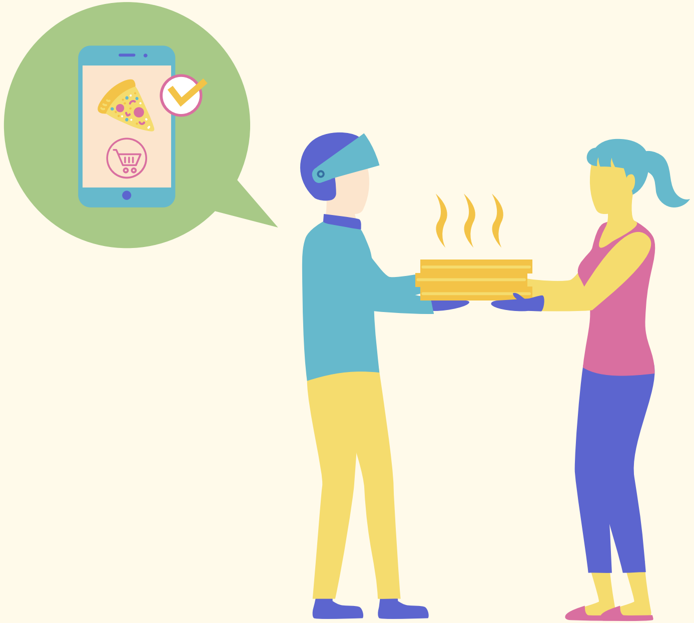
```
--------------------------------
   Total Cost:   1500
   Minutes:   25.0
--------------------------------
depart Downtown Berkeley, 0, 0
red Downtown Berkeley, 0, 0
red Ashby, 180, 180
red MacArthur, 240, 420
red 19th Street, 180, 600
red 12th Street, 120, 720
red West Oakland, 300, 1020
red Embarcadero, 420, 1440
red Montgomery Street, 60, 1500
arrive Montgomery Street, 0, 1500
```



Shortest Path Algorithm

- However, we find that when we apply it to the BART system, most of the routes from Downtown Berkeley to our determined delivery hubs only required one transit line with no need to transfer.

- In the future, you would run this algorithm again if a station ever becomes inaccessible, or when the BART system becomes more complicated. Similar process can be applied to other transit systems.



Other NoSQL Database Uses

Redis

- Store and update live delivery data

- Store/update inventory data

- Store server side cookies on web and API servers

MongoDB

- Store Bart routes to delivery hubs

- Web server and API server

- Store multiple POV’s of historical delivery data

Thanks!

Questions?