Jacob Martin
DSBA 6520 Network Science
8/9/2021
Final Project

San Francisco Bay Bike Share Data Analysis

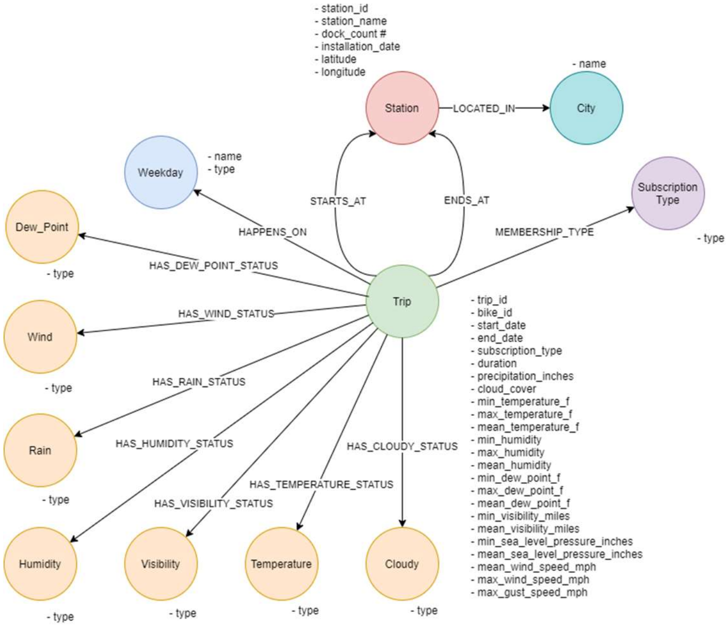
Dataset & Business Use Case

The data set chosen comes from Kaggle.com and is representative of a bike share program in the San Francisco Bay Area. It spans from bike trips from August 2013 until August 2015. The data set includes information pertaining to bike stations, individual bike trips, and weather data in 4 different csv files. Over the period there are a total of 670,000 trips recorded. Each trip has a start and ending station, duration, if the individual was a subscriber, and the bike's id. The weather data consists of temperature, dew point, humidity measures, and other values pertaining to the day's weather in each of the cities.

Our goal is to create an analysis of the data to answer some basic questions about the current ridership. These include understanding what bike stations are the most and least important in the network. It will also be useful to know if all the stations create a single network or if the network is segmented in multiple components. There may be conditions such as rivers or highways which naturally divide the ability to get a bike between stations. Understanding issues like these could aid in developing a more robust network as well as knowing what makes stations better than others. After profiling the data, it shows that 15% of rides are by customers and 85% are by subscribers. We will investigate if certain stations are responsible for a disparity between those types.

Even though weather data is included, it will not be used in the graph algorithms used later. The weather in the SF Bay area is very preferable year-round for outdoor cycling. Thus, the variation in our data is small compared to other factors. As will be shown later, the number one factor in determining ridership per day is whether it is a weekday or weekend. Holidays will play their role as well. We have decided to focus on understanding the existing network for creation of stations.

Graph Data Model

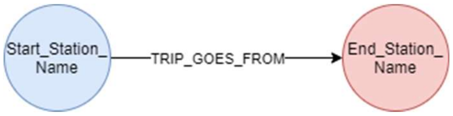


The Graph model is centric around individual trips. Stations share common locations so the start and end stations will be linked to common nodes for latitude, longitude, and other properties. Stations are connected to the trip id by the STARTS_AT and ENDS_AT relationships. Our weather data is not granular to the time of the trip but rather to the day the trip was taken. This makes prediction based on the weather data hard to execute reliably because there is not any way to line it up exactly with a given trip. Thus, we created nodes for each of the weather types. For temperature, dew point, and humidity we created types of high, medium, and low using the data from 2014 as our full year fluctuation. As will be explained later, only the data from 2015 will be used in our analysis due to computational constraints so the 2014 data made a full year's worth of information to create box plots on. Divisions were drawn in the data at

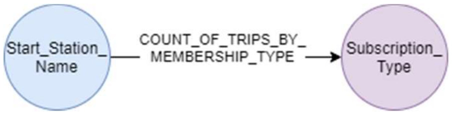
25% and 75% to create the three sections. For wind, rain, cloud cover, and visibility a division was drawn at a reasonable level to form two groups. Judgement was used based on the code guide to determine each section.

Graph Projections

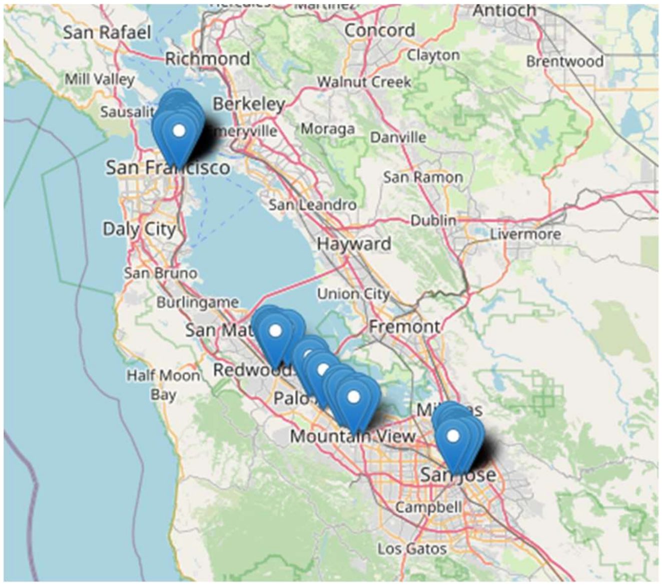
Our first projection from our graph model is to help us learn more about where individuals are going. We have a bipartite projection from the start station to the ending station that is linked through the trip ID node.

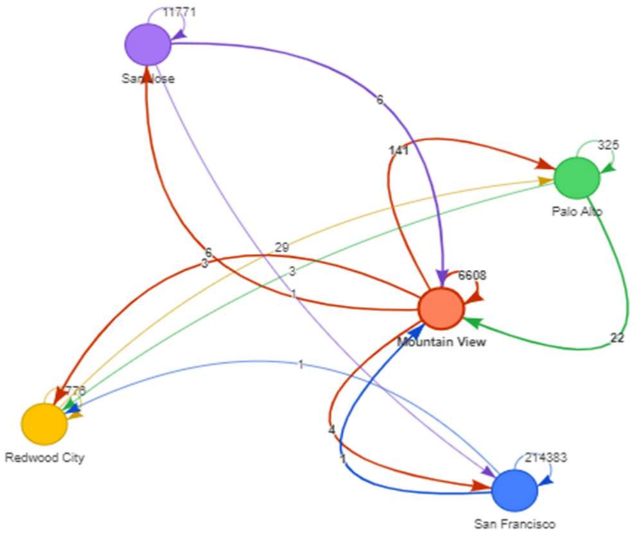


As mentioned previously, 85% of trips are taken by subscribers. Understanding what stations users typically begin their ride at can aid in understanding our users' habits. This projection goes from the start station to the subscription type based on a count of the trip id field.



Exploratory Analysis





The data is very clearly divided into 3 clusters of information. Each cluster is easily visible from a geographic perspective. San Francisco is approximately 50 miles from San Jose and this distance creates a natural barrier for trips to take place between cities. San Francisco makes up the largest cluster. It is followed by San Jose as the second largest. The 3 cities of Mountain View, Palo Alto, and Redwood City are the only cities that have a significant number of trips between them compared to the overall number starting in the city. The graph above shows the number of trips taken from one city to another with the arrow of direction pointing from the start city to the end city.

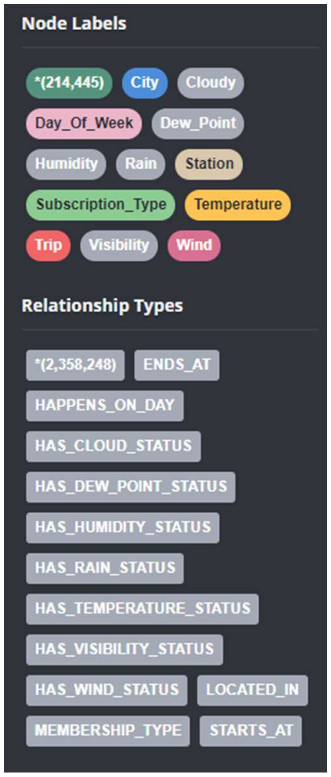
Because most of the data is isolated to each individual cluster, we narrowed down the data we will be using for the analysis to include only the San Francisco trips. Without running a graph algorithm, it is easy to see we have strongly connected components inside of a city, but not from city to city. We also limited the data to trips that happen in 2015.

The data ends in August of 2015 which provides us with 8 months and 214,383 trips. This was done due to the computational complexity of the algorithms and data upload. When trying to use a larger subset of the data, there were issues with Neo4j rejecting our information due to lack of memory.

The average trip duration is 15.13 minutes which shows that a typical trip is local. Using google maps to create a route distance, the longest route in the San Francisco cluster is between Embarcadero at Sansome and Townsend at 7th. They are 3.5 miles by path which should take approximately 18-20 minutes.

Neo4J Database Setup

Our completed set up of the Neo4J graph database consists of 214,445 nodes. Of that, 214,383 are trips, 2 subscription nodes, 1 city node, 35 stations, 7 day of week nodes, and 17 different types of weather nodes. Using the relationships from our schema above, we have 2,358,248 relationships. The image to the right shows the node labels and relationship types.



Node Labels

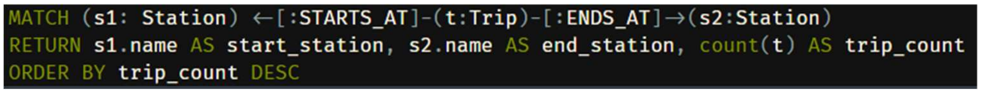
*(214,445) City Cloudy Day_Of_Week Dew_Point Humidity Rain Station Subscription_Type Temperature Trip Visibility Wind

Relationship Types

*(2,358,248) ENDS_AT HAPPENS_ON_DAY HAS_CLOUD_STATUS HAS_DEW_POINT_STATUS HAS_HUMIDITY_STATUS HAS_RAIN_STATUS HAS_TEMPERATURE_STATUS HAS_VISIBILITY_STATUS HAS_WIND_STATUS LOCATED_IN MEMBERSHIP_TYPE STARTS_AT

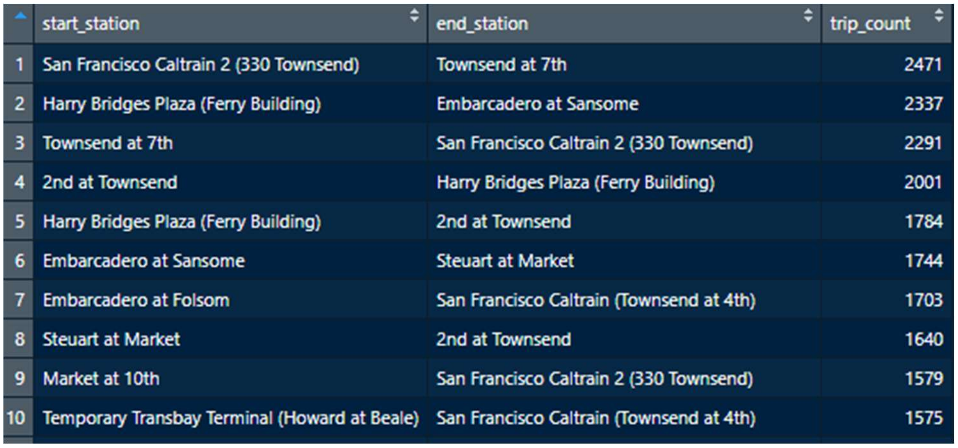
Cypher Queries

1.



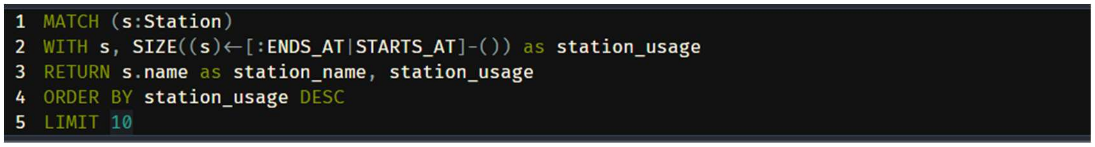
```
MATCH (s1: Station) ←[:STARTS_AT]-(t:Trip)-[:ENDS_AT]→(s2:Station)
RETURN s1.name AS start_station, s2.name AS end_station, count(t) AS trip_count
ORDER BY trip_count DESC
```

Our first Cypher query is designed to give us an idea of the station-to-station connectivity. This is useful for understanding which trip start and destinations are the most popular. It displays the station the trip started at and the station they end at. In our data, it is important to remember that all stations are connected to all other stations. We do not know how the rider got there or if they stopped along the way. Information like this can aid in ensuring the supply of bikes at each station can keep up with demand. Since our stations have sensors that can tell how many bikes are currently available at a given station, we can use this information to ensure that these stations receive priority to keep them supplied. The chart below shows the top 10 trips. This data will also be used as a fold for our future graph algorithms with the number of trips as the weight.



	start_station	end_station	trip_count
1	San Francisco Caltrain 2 (330 Townsend)	Townsend at 7th	2471
2	Harry Bridges Plaza (Ferry Building)	Embarcadero at Sansome	2337
3	Townsend at 7th	San Francisco Caltrain 2 (330 Townsend)	2291
4	2nd at Townsend	Harry Bridges Plaza (Ferry Building)	2001
5	Harry Bridges Plaza (Ferry Building)	2nd at Townsend	1784
6	Embarcadero at Sansome	Steuart at Market	1744
7	Embarcadero at Folsom	San Francisco Caltrain (Townsend at 4th)	1703
8	Steuart at Market	2nd at Townsend	1640
9	Market at 10th	San Francisco Caltrain 2 (330 Townsend)	1579
10	Temporary Transbay Terminal (Howard at Beale)	San Francisco Caltrain (Townsend at 4th)	1575

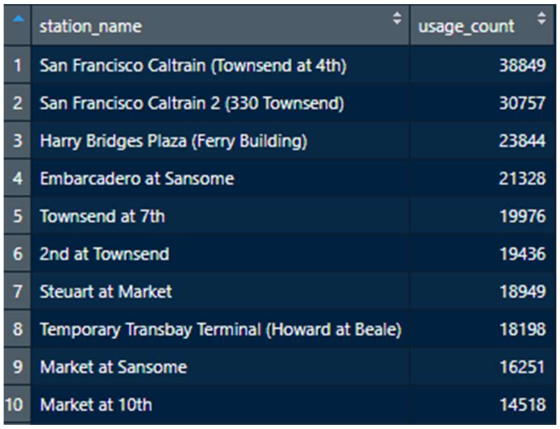
2.



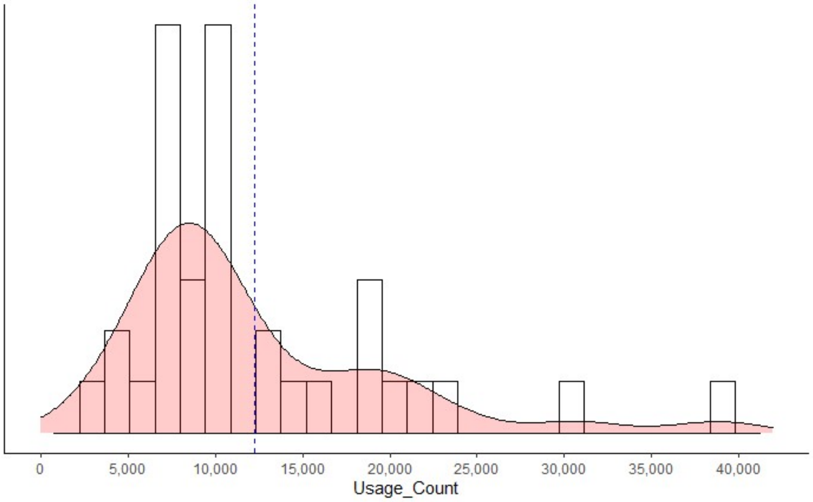
```
MATCH (s:Station)
WITH s, SIZE((s)←[:ENDS_AT|STARTS_AT]-()) as station_usage
RETURN s.name as station_name, station_usage
ORDER BY station_usage DESC
LIMIT 10
```

Similarly, we can perform a query to identify the most used stations. Our query calculates the degree for the relationships of ‘STARTS_AT’ and ‘ENDS_AT’. Effectively, we are calculating the usage of an individual station. It includes all trips that go through the station. This information can be used similarly to the prior query where we can understand demand mechanics for a station. Additionally, this data can be used to understand the maintenance needs of a station.

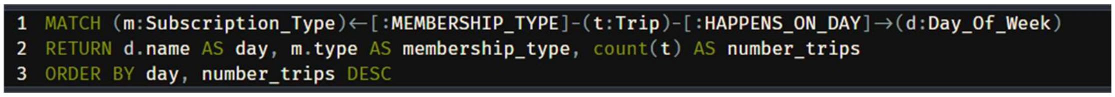
If we estimate that bike terminals break down after a certain number of bike dockings, this information could be used to estimate a predictive maintenance model. Below are the top 10 stations and a distribution of all the stations. The density plot shows a significant disparity in the usage between the most used stations vs the least used ones. The mean value is plotted as the blue dotted line at 12,250. An interesting note about the top two stations is that they are geographically right next to each other. In the data they are labeled as two separate stations, but geographically they are within 50 feet proximity. Zooming in to that location shows that the stations are both at a major train station in the heart of San Francisco. It is likely that they have high traffic due to commuters riding into the city by train, then using the bike system to travel the last mile. If both stations are tallied together, they far exceed the next highest used bike station.



	station_name	usage_count
1	San Francisco Caltrain (Townsend at 4th)	38849
2	San Francisco Caltrain 2 (330 Townsend)	30757
3	Harry Bridges Plaza (Ferry Building)	23844
4	Embarcadero at Sansome	21328
5	Townsend at 7th	19976
6	2nd at Townsend	19436
7	Steuart at Market	18949
8	Temporary Transbay Terminal (Howard at Beale)	18198
9	Market at Sansome	16251
10	Market at 10th	14518

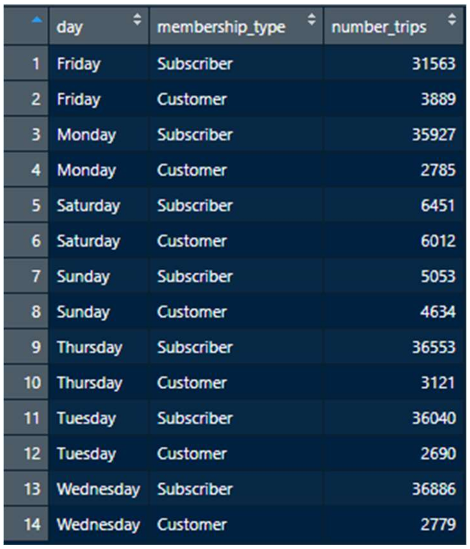


3.

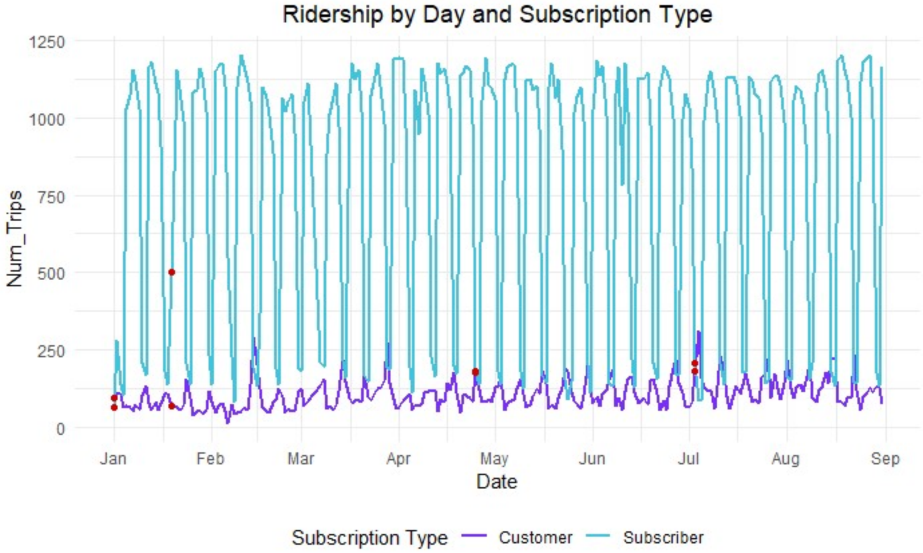


```
MATCH (m:Subscription_Type)←[:MEMBERSHIP_TYPE]-(t:Trip)-[:HAPPENS_ON_DAY]→(d:Day_Of_Week)
RETURN d.name AS day, m.type AS membership_type, count(t) AS number_trips
ORDER BY day, number_trips DESC
```

Day of the week is a strong factor in prediction of the number of trips taken daily. With subscribers, ridership is higher on the weekdays. With customers, it is inverse. The chart below details the total number of trips taken on a given day by each group. The graph plots the number of trips taken each day and line color is used for each group. Holidays are plotted as red dots. This query gives us information about trends in ridership. Since subscribers provide the primary usage of our stations, we can schedule maintenance on equipment at low peak times.

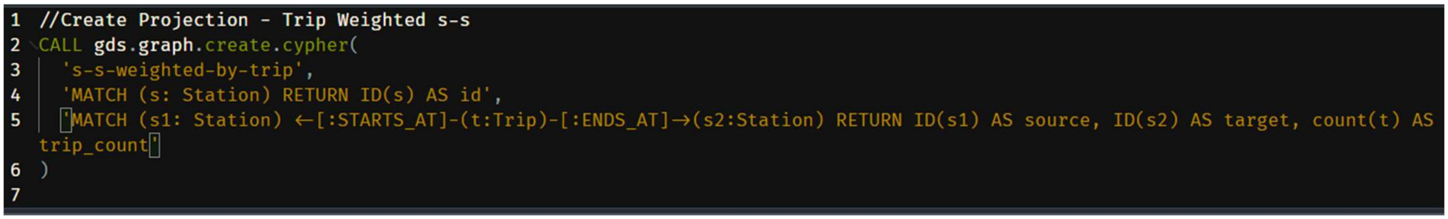


	day	membership_type	number_trips
1	Friday	Subscriber	31563
2	Friday	Customer	3889
3	Monday	Subscriber	35927
4	Monday	Customer	2785
5	Saturday	Subscriber	6451
6	Saturday	Customer	6012
7	Sunday	Subscriber	5053
8	Sunday	Customer	4634
9	Thursday	Subscriber	36553
10	Thursday	Customer	3121
11	Tuesday	Subscriber	36040
12	Tuesday	Customer	2690
13	Wednesday	Subscriber	36886
14	Wednesday	Customer	2779

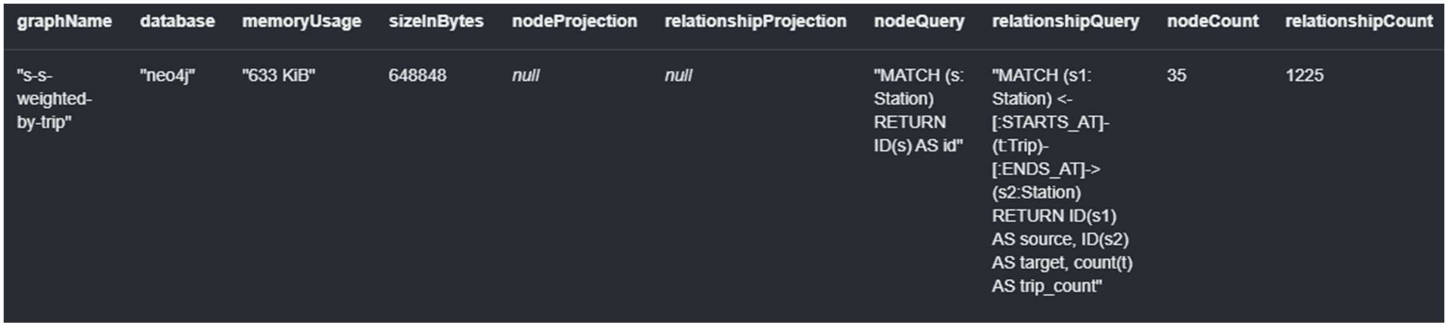


Graph Algorithms

1. Weakly Connected Components on Start to End Station Trip Data



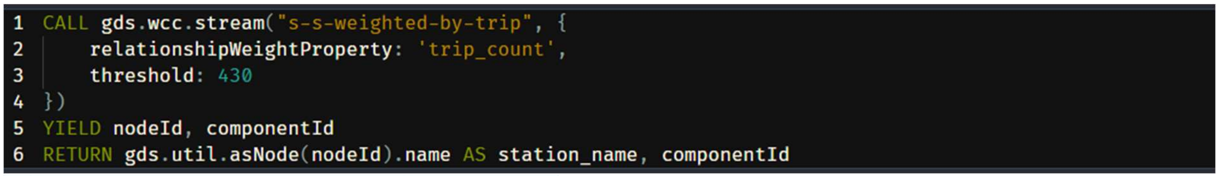
```
//Create Projection - Trip Weighted s-s
CALL gds.graph.create.cypher(
  's-s-weighted-by-trip',
  'MATCH (s: Station) RETURN ID(s) AS id',
  'MATCH (s1: Station) ←[:STARTS_AT]-(t:Trip)-[:ENDS_AT]→(s2:Station) RETURN ID(s1) AS source, ID(s2) AS target, count(t) AS trip_count'
)

```



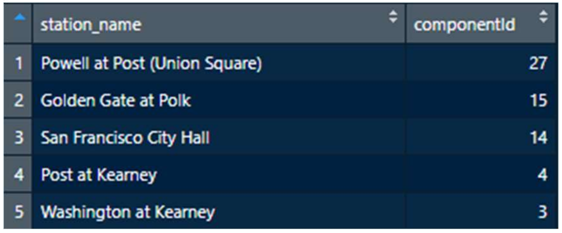
graphName	database	memoryUsage	sizeInBytes	nodeProjection	relationshipProjection	nodeQuery	relationshipQuery	nodeCount	relationshipCount
"s-s-weighted-by-trip"	"neo4j"	"633 KiB"	648848	*null*	*null*	"MATCH (s: Station) RETURN ID(s) AS id"	"MATCH (s1: Station) <-[:STARTS_AT]-(t:Trip)-[:ENDS_AT]->(s2:Station) RETURN ID(s1) AS source, ID(s2) AS target, count(t) AS trip_count"	35	1225

First, we need to set up our graph projection using the above code segment. It spans from station to station with the number of trips taken between stations as the weight. Then we can perform our weakly connected components algorithm. Since all stations are connected by some number of trips taken between them, we need to set a threshold. Our threshold is set at 430, which uses a 2x coefficient for the distance of our box plot whisker on the high end. To gain meaningful information the threshold needs to be set rather high. Our results show that 5 stations return as weakly connected. 4 of these stations are also among the lowest ridership over all when aggregating by trips starting at the station. The one exception to that is Powell at Post (Union Square).

The graph algorithm is helpful for understanding which components have a poor connection to other stations in the set. After applying the threshold, these 5 stations were found to have no connection to the main body of stations or each other. This is important for understanding ridership. The information can be coupled with external data to provide insights on what makes a location good or bad for future stations. If we treat a station as a gateway destination to nearby buildings and entertainment, we can model how riders are moving through the city.

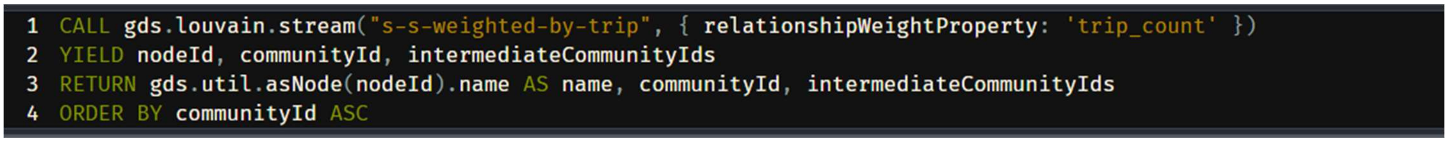


```
CALL gds.wcc.stream("s-s-weighted-by-trip", {
    relationshipWeightProperty: 'trip_count',
    threshold: 430
})
YIELD nodeId, componentId
RETURN gds.util.asNode(nodeId).name AS station_name, componentId
```

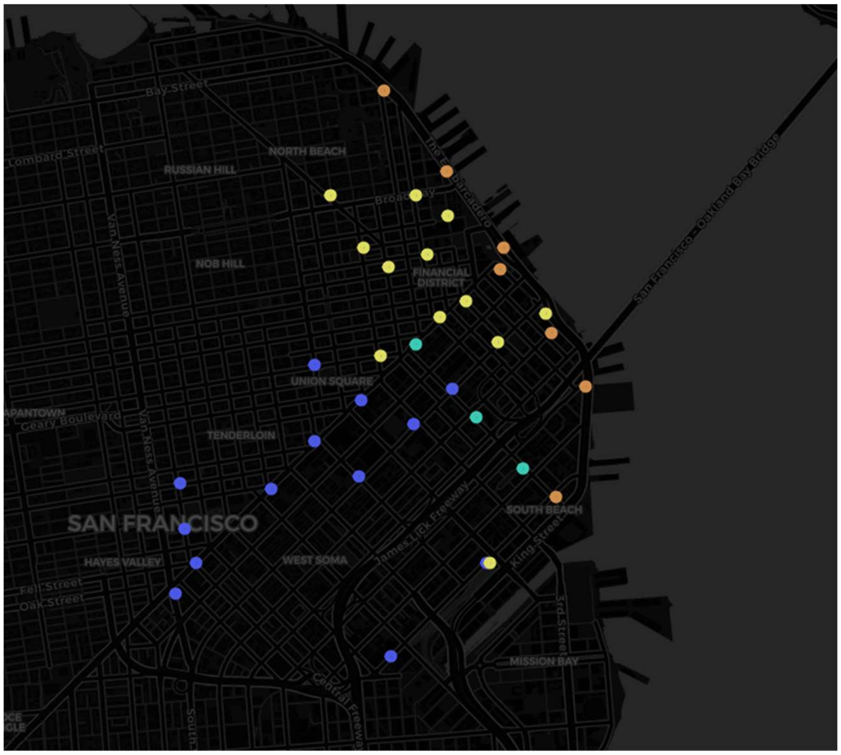


	station_name	componentId
1	Powell at Post (Union Square)	27
2	Golden Gate at Polk	15
3	San Francisco City Hall	14
4	Post at Kearney	4
5	Washington at Kearney	3

2. Louvain Clustering on Start to End Station Weights by Total Number of Trips



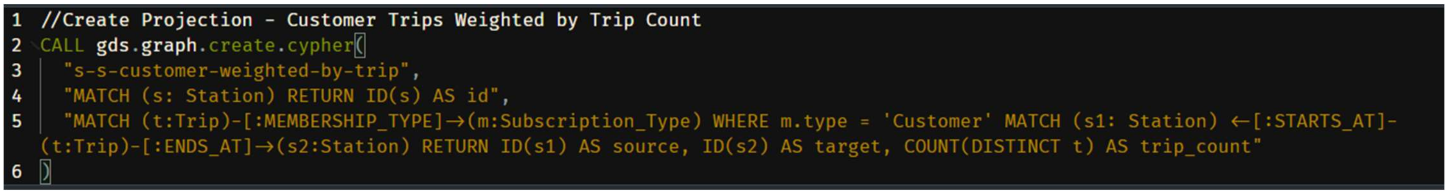
```
CALL gds.louvain.stream("s-s-weighted-by-trip", { relationshipWeightProperty: 'trip_count' })
YIELD nodeId, communityId, intermediateCommunityIds
RETURN gds.util.asNode(nodeId).name AS name, communityId, intermediateCommunityIds
ORDER BY communityId ASC
```



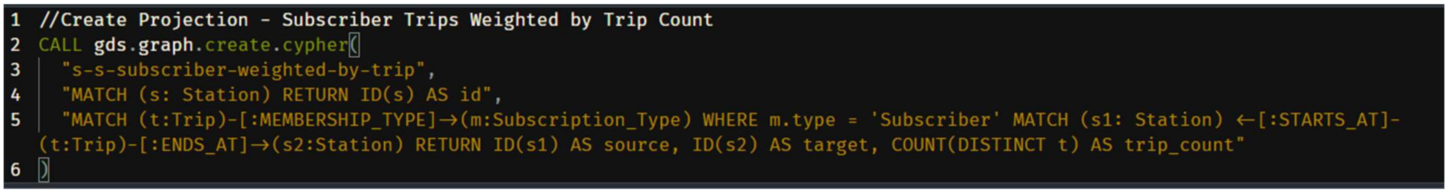
This graph tells us information about clusters in the trips riders are taking. It yields 4 communities in our San Francisco stations and no intermediate communities. We can plot the communities on a map to understand them from a geographical perspective. Communities are projected as color in this visualization. The smallest community is only 3 stations and the largest is 13. A map like this is helpful because it gives insight as to where riders typically travel. As discussed earlier, the two most frequently used stations are at the major train station. One of those stations is in each of the blue and yellow clusters. The yellow community seems to surround the financial district. This could be individuals coming into the city by train and then riding to their office. This is feasible considering we know rides for subscribers follows the day of the week. The blue stations surround the streetcar line which could take people to a method of traversing the city.

3. Page Rank on Customer and Subscriber Trips

First, we set up two new graph projections that encapsulate each subset of the data for subscription type. Next, we ran page rank on each set of the data to determine centrality importance. This is helpful for understanding the difference between the way different users interact with our network of bike stations.

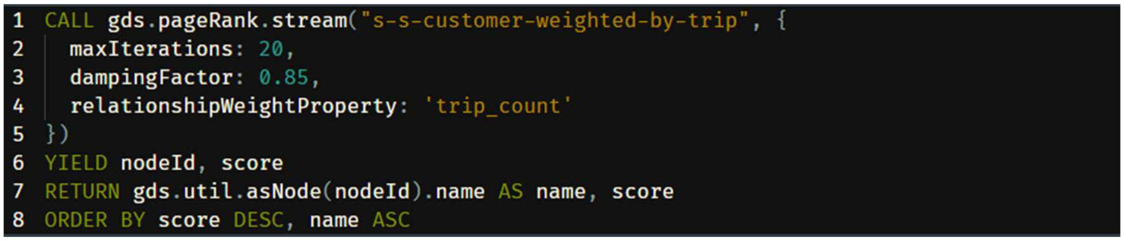


```
//Create Projection - Customer Trips Weighted by Trip Count
CALL gds.graph.create.cypher(
  "s-s-customer-weighted-by-trip",
  "MATCH (s: Station) RETURN ID(s) AS id",
  "MATCH (t:Trip)-[:MEMBERSHIP_TYPE]→(m:Subscription_Type) WHERE m.type = 'Customer' MATCH (s1: Station) ←[:STARTS_AT]-(t:Trip)-[:ENDS_AT]→(s2:Station) RETURN ID(s1) AS source, ID(s2) AS target, COUNT(DISTINCT t) AS trip_count"
)
```



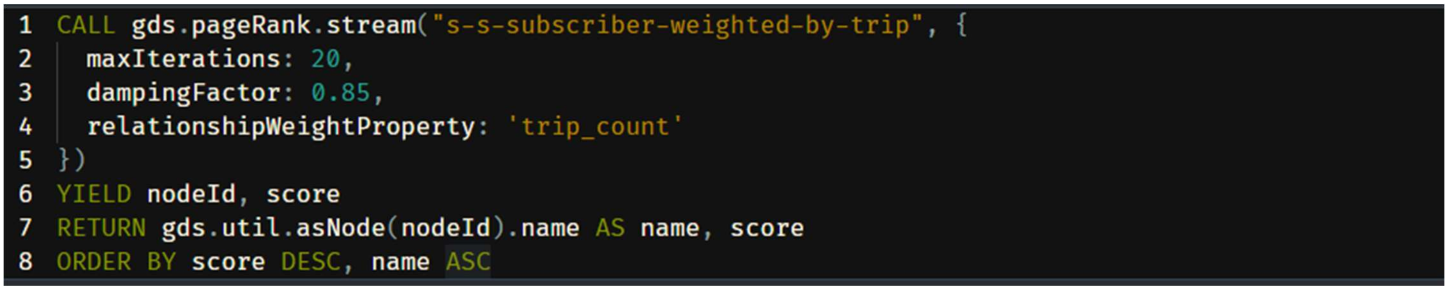
```
//Create Projection - Subscriber Trips Weighted by Trip Count
CALL gds.graph.create.cypher(
  "s-s-subscriber-weighted-by-trip",
  "MATCH (s: Station) RETURN ID(s) AS id",
  "MATCH (t:Trip)-[:MEMBERSHIP_TYPE]→(m:Subscription_Type) WHERE m.type = 'Subscriber' MATCH (s1: Station) ←[:STARTS_AT]-(t:Trip)-[:ENDS_AT]→(s2:Station) RETURN ID(s1) AS source, ID(s2) AS target, COUNT(DISTINCT t) AS trip_count"
)
```

PageRank on Customer Projection:



```
CALL gds.pageRank.stream("s-s-customer-weighted-by-trip", {
  maxIterations: 20,
  dampingFactor: 0.85,
  relationshipWeightProperty: 'trip_count'
})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS name, score
ORDER BY score DESC, name ASC
```

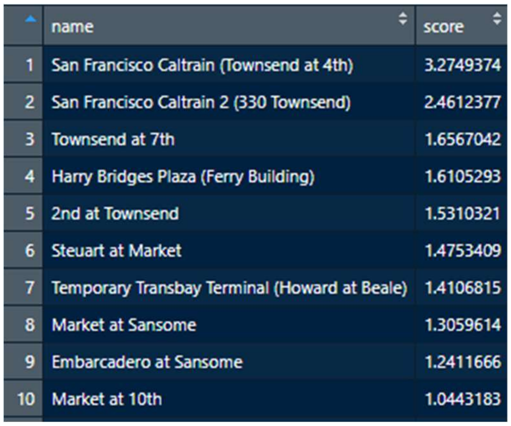
PageRank on Subscriber Projection:



```
CALL gds.pageRank.stream("s-s-subscriber-weighted-by-trip", {
  maxIterations: 20,
  dampingFactor: 0.85,
  relationshipWeightProperty: 'trip_count'
})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS name, score
ORDER BY score DESC, name ASC
```

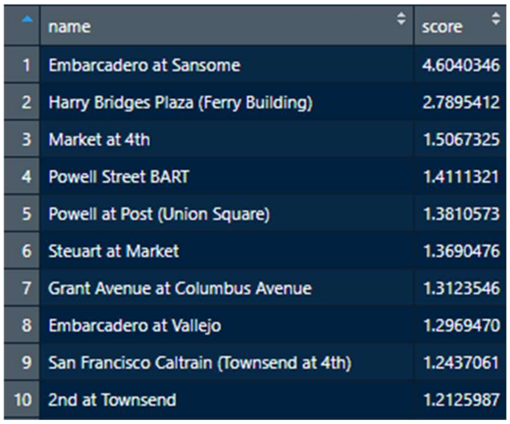
The resulting stations are quite different in their importance to the network. Understanding the difference between how customers and subscribers use the network will aid in how the bike share program chooses new locations for stations. Since each group uses the network slightly different, it may be necessary to issue preference to one group over the other during this process. To ensure bikes are stocked for peak times (weekdays for subscribers and weekends for customers), we can use node centrality through this algorithm for estimating demand.

Subscribers

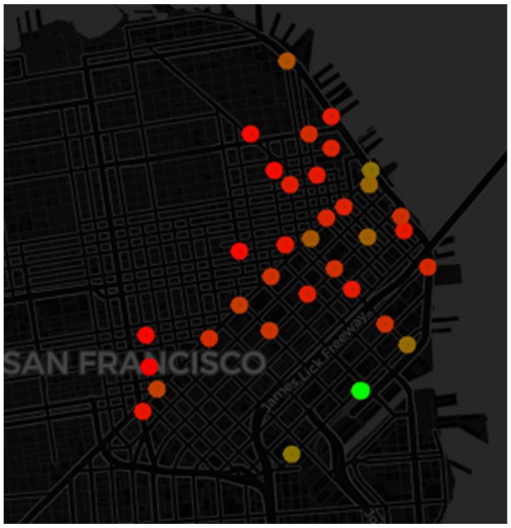


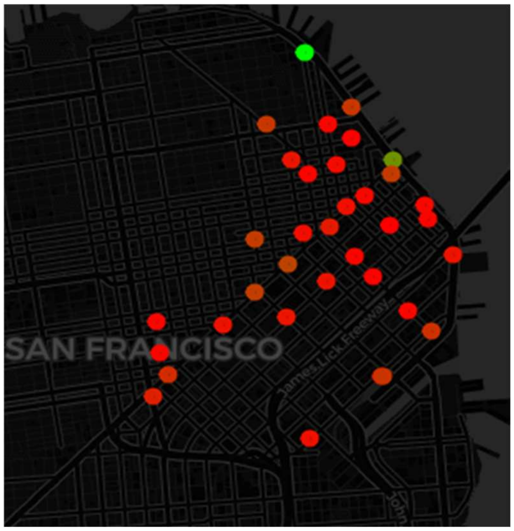
	name	score
1	San Francisco Caltrain (Townsend at 4th)	3.2749374
2	San Francisco Caltrain 2 (330 Townsend)	2.4612377
3	Townsend at 7th	1.6567042
4	Harry Bridges Plaza (Ferry Building)	1.6105293
5	2nd at Townsend	1.5310321
6	Steuart at Market	1.4753409
7	Temporary Transbay Terminal (Howard at Beale)	1.4106815
8	Market at Sansome	1.3059614
9	Embarcadero at Sansome	1.2411666
10	Market at 10th	1.0443183

Customers



	name	score
1	Embarcadero at Sansome	4.6040346
2	Harry Bridges Plaza (Ferry Building)	2.7895412
3	Market at 4th	1.5067325
4	Powell Street BART	1.4111321
5	Powell at Post (Union Square)	1.3810573
6	Steuart at Market	1.3690476
7	Grant Avenue at Columbus Avenue	1.3123546
8	Embarcadero at Vallejo	1.2969470
9	San Francisco Caltrain (Townsend at 4th)	1.2437061
10	2nd at Townsend	1.2125987



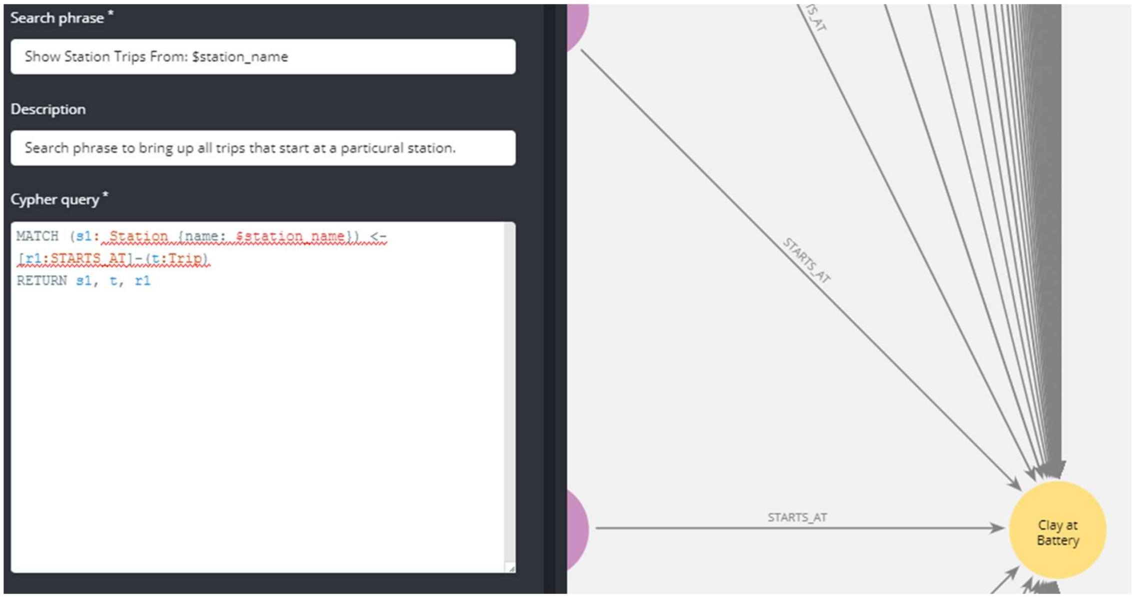


Conclusion

Building a bike network requires intuition about where to build new stations and how to maintain the existing network. Accessing the connectivity of the existing structure through the routes of each individual trip can provide insight as to the individual value of a given station. This can be useful for providing preferential and predictive maintenance. When building new stations, it may require that they be placed to benefit certain groups of riders over others. In the case of the existing San Francisco bike network, it has aspects that work for customers and subscribers. Subscribers are the majority users of the network with strong emphasis on node centrality at the train station. Customers tend towards centrality of stations on the north side of the city near the ferry.

Cypher Actions

1. Our first Cypher Action is a search to bring up all trips that start from a given station. We also created another action that performs the same query on the [:ENDS_AT] relationship. These actions give an analyst the ability to quickly see all trips pertaining to a given station. From there they can expand trip nodes to explore information they are interested in. This may be the connection of different weather conditions from the node or where the end station is located.



Search phrase *

Show Station Trips From: $station_name

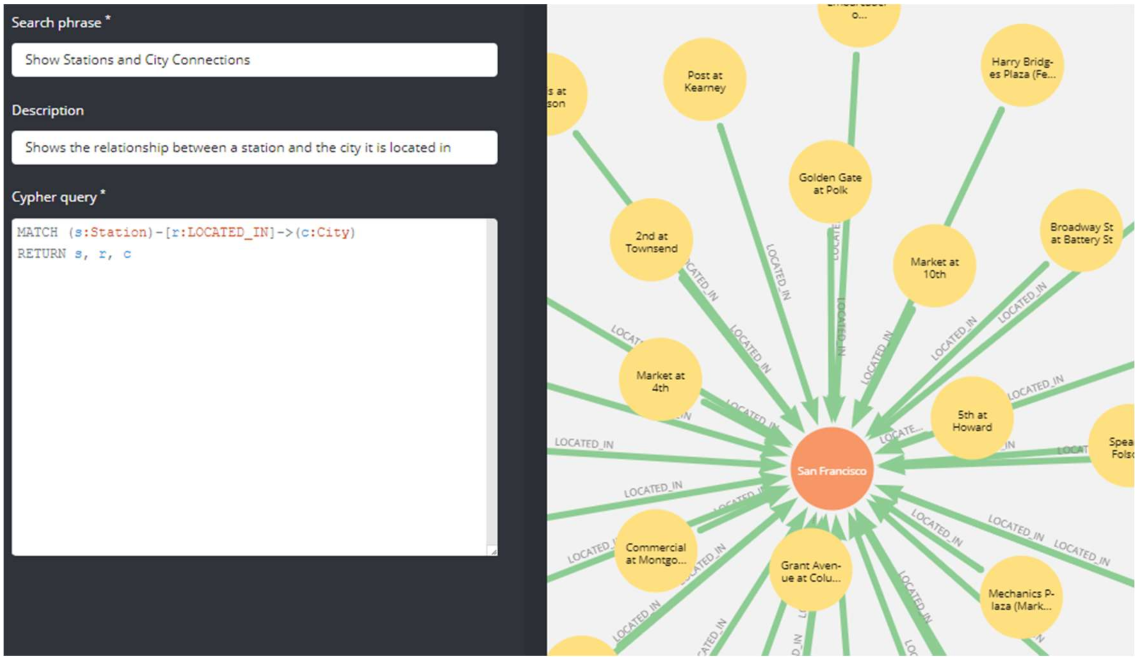
Description

Search phrase to bring up all trips that start at a particural station.

Cypher query *

```
MATCH (s1: Station {name: $station_name}) <-
[r1:STARTS_AT]-(t:Trip)
RETURN s1, t, r1
```

2. We also created another cypher query that brings up the relationship between stations and what city they are located in. This query doesn't hold a lot of value as the current graph data base is set up. As mentioned previously, we had to narrow down our database due to the computational complexity of it. The data base currently only contains trips that started and ended in San Francisco. Meaning that when running the query there will only be 1 city node. The cypher action will be useful in a deployment model that contains the entire bike network.



Search phrase *

Show Stations and City Connections

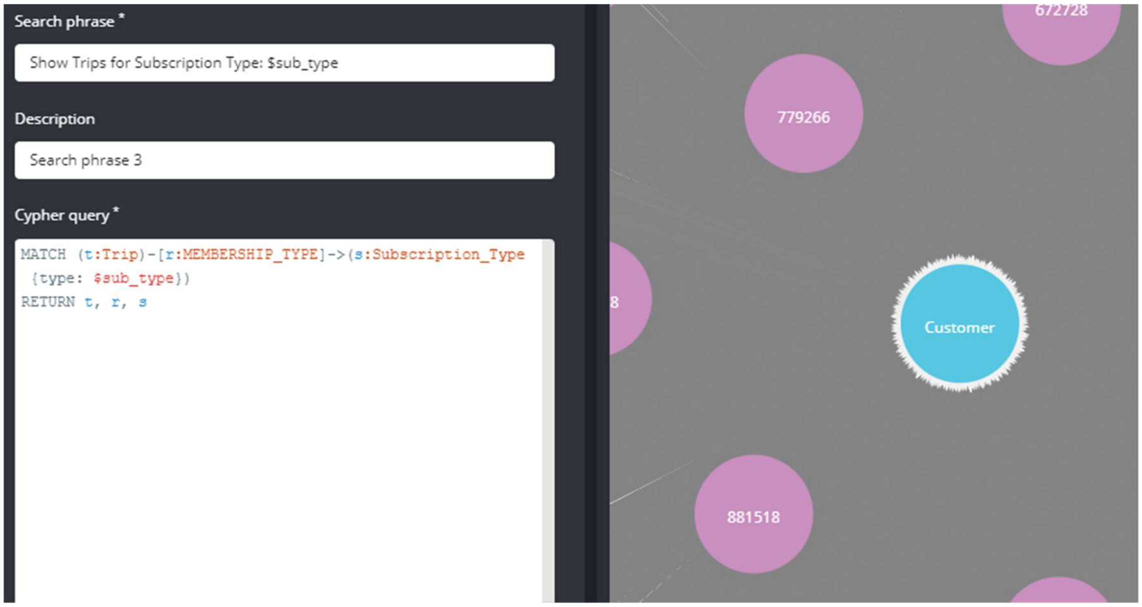
Description

Shows the relationship between a station and the city it is located in

Cypher query *

```
MATCH (s:Station)-[r:LOCATED_IN]->(c:City)
RETURN s, r, c
```

3. Lastly, we created a search to show the connections of trips to each subscription type. If an analyst is interested in understanding what trips have a given membership type they have the ability to run a quick cypher action that expands this relationship. It is based upon the input value $sub_type, which represents a Subscription_Type node's type property.



Search phrase *

Show Trips for Subscription Type: $sub_type

Description

Search phrase 3

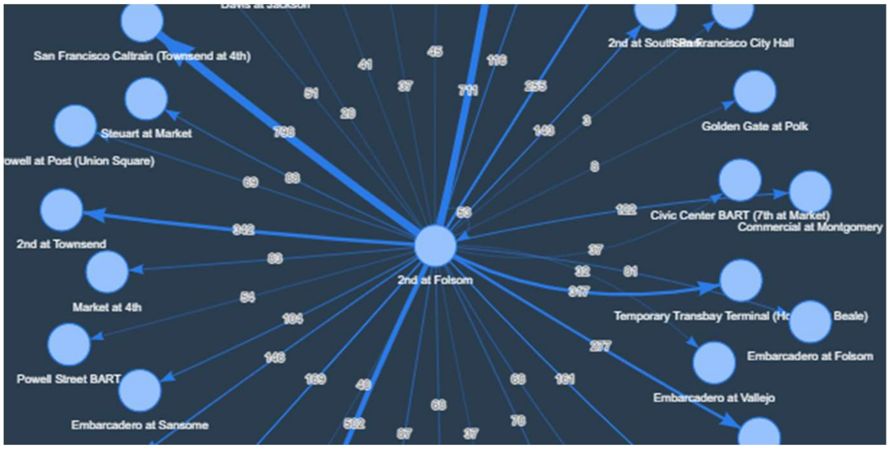
Cypher query *

```
MATCH (t:Trip)-[r:MEMBERSHIP_TYPE]->(s:Subscription_Type
 {type: $sub_type})
RETURN t, r, s
```

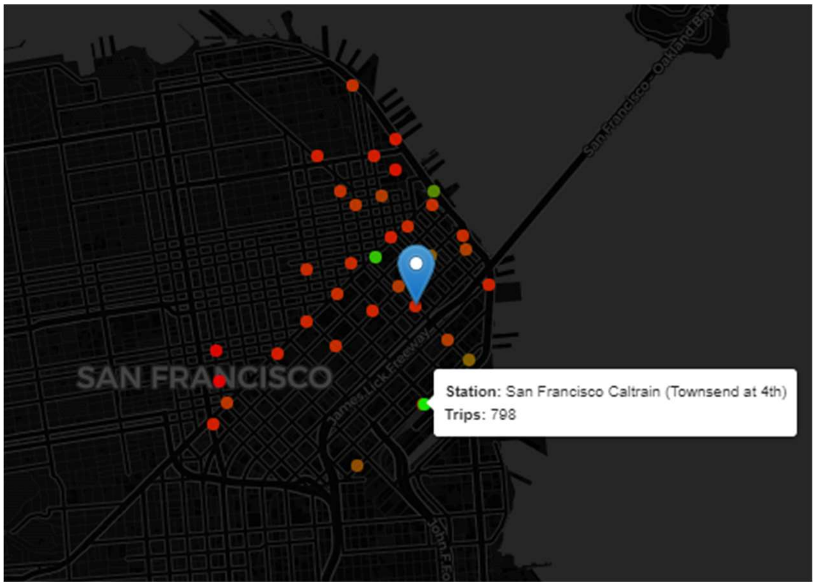
GUI Dashboard

The main issue with Neo4j's Bloom product is that it can't visualize the data on a projection fold. That fault makes using many of the cypher queries above ineffective. Much of how our data is structured is only useful when we aggregate values upon a projection. This is because the data hinges on the trip nodes and any individual trip doesn't hold a lot of value. It is when we aggregate across many that trends appear in the data. Therefore, we created a dashboard that represents the fold of trip journeys tallied between stations (Link on the last page).

At the top of the page are options on how the user can filter their data. It requires a station where trips start. Options are available for subscription type, week day/end, weather conditions, and a date slider. These give the user flexibility to explore the data as needed for their analysis. There are 2 viewing options for how the visualization is produced. The network view shows each station as a node and uses a line between them to indicate the relationship of the trip going from one station to another. Line thickness is used as a dimension of number of trips. The second view utilizes a map to give the user a geographic understanding of the data. If we think of there being a cost to travel to another station (The distance between them), this view helps the user understand some of the clustering that was discussed earlier. Color is used as the weight dimension in this view. Green represents many trips and red represents few. Each view is standardized based on the data currently selected. By hovering over a station, the user can gather additional details.



At the bottom of the dashboard is a table that shows all rows of the data returned. This feature gives the user the ability to explore at a higher level of data granularity.


SAN FRANCISCO
Station: San Francisco Caltrain (Townsend at 4th)
Trips: 798
James Lick Freeway
San Francisco – Oakland Bay

Future Work

1. Integrate our network model into a phone application for ridership information

Using a phone application, customers and subscribers will have the ability to utilize the bike share network. It can work by allowing them to rent bikes. As an added layer, a model could be built to predict locations near bike stations that they may be interested in. San Francisco has a wide array of public data available, including neighborhood, business, and retail development information (Example: https://data.sfgov.org/Economy-and-Community/Map-of-Registered-Businesses-San-Francisco/9tqe-vdng). By incorporating popular destinations along common routes, we could suggest to riders stops and what the closest station is that stop.

2. Ask additional questions of our data

Because this project is mainly related to network analysis of the existing bike network, we have focused on answering questions about connectivity. There are many other questions that we could dig into with the data:

- Is there a minimum distance that we should separate stations?
- Do riders prefer to walk/drive if their destination is less/more than a certain number of blocks from their current location?
- Are trends in the data for station-to-station travel fixed for weather conditions?
- Are stations with connected designated bike lanes more popular?
- What stores/destinations/public transit/etc. are near popular stations?

Summary of Work

Public services can be a very complex. In our study of the data, we have discovered trends that may help the bike network in its future expansion of where and why to build stations. This body of work shows how the bike network may have to decide which groups of people they are going to serve when doing so. Additionally, we have created some practical applications for machine learning in the data to suggest locations along potential bike paths to increase the user experience. The network model and cypher queries used can benefit San Francisco's bike share program as it expands.

Appendix

1. Anecdotal Challenges Along the Way

- One frustration that I had with this project is that the current package that connects R and Neo4j is nonfunctional with Neo4j version 4+. This limited my ability to perform a lot of the cypher queries from R Studio and then push those into visualizations. Looking back, if I had known that up front I would have switched and done the work in Python. I was really impressed by the gentleman who had live cypher queries of New York City crime data and had something similar in mind for my own final project deliverable. Due to the limitation, I ended up having to build the application off flat data.
- Learning Cypher was a very interesting experience. It is certainly an intuitive way to instruct the computer to yield the information desired, but I felt like there were things that became more difficult. One example of this was trying to aggregate data in certain ways by groups of information. While on flat data in R this would take 1 line of code, it took much more effort to draw the same information out of Neo4j. My experience here was that if I want to do data cleaning or data investigation, R was the way to go. I'm excited to learn more about network machine learning models in Neo4j and hope to do more on that front after the class is finished. They seem very powerful and what the real value of using Cypher/Neo4j are.

2. Links

Dataset: https://www.kaggle.com/benhamner/sf-bay-area-bike-share

Github: https://github.com/Troxoboxo/SF_Bike_Share

GUI Dashboard: https://jmart230.shinyapps.io/BikeProj_Shiny/

Bike Share Website: https://www.sfmta.com/getting-around/bike/bike-share

** Note the Github holds all code used for Neo4j implementation, R Shiny app development, and cleaning steps done in R