TransAnalytics

Transportation Analytics Center of Excellence for the City of Chicago

Executive Team Presentation

Impact of Transportation Alternatives on City Transportation

December 12th 2020

Akhil Ranjan, Jingyu Zhang, Olga Niyibizi, Sahil Sachdev

Agenda

ONE

Overview

—

TWO

Data Profile

—

THREE

Methodology

—

FOUR

Platform

—

FIVE

Insights

—

SIX

Conclusion

—

ONE

Overview

Executive Summary

The usage of e-scooters is quickly emerging as a popular method of transportation in various metropolitan cities.

In the summer of 2019 the city of Chicago launched their own pilot program to track rider usage.

With the growing interest in this method of transportation, it is important for the city to determine if they want to invest and allow companies to operate these e-scooters in Chicago.

Business Use Case

Analyze the E-scooter pilot program in addition to other methods of transportation and their impacts to better prepare governmental agencies when planning transportation infrastructure for their communities.

TWO

Data Profile

Data Profile

	E-Scooter Dataset	Public Transportation Dataset	Divvy Bike Dataset	Ridership for Taxi Usage
Data Size	~194 MB	~30 MB	~5 GB	~30 GB
Number of Observations	~711,000	~33,800	~21M	~194 Million
Types of Data Used	Structured	Structured	Structured	Structured

Data Profile

Data Profile Continued:

- ❖ Data sourced from the website *the City of Chicago*
- ❖ Data Clean-up (Filtering, restructuring)
  - ➢ Only focused on data from June 15, 2019 to October 15, 2019 and June 15, 2018 to October 15, 2018
- ❖ Integration of data
  - ➢ Taxi, Divvy, and E-scooter

THREE

Methodology

Tools Overview

Data Design Steps

Ingestion and Cleanup | Storage | Delivery and Insights

Data Cleaned using Open Refine, Python.
Web scrape the data from City of Chicago (data.cityofchicago.org)

Database created using MySQL.

Used GCP for storing the DDL & DML tables and sharing it among the team mates.

Reports and insights generated by using Tableau

Data Quality Dimensions

Data Quality

1. Completeness
Filtered the data from June 15, 2019 to October 15, 2019 and June 15, 2018 to October 15, 2018 for the dats

2. Consistency
Synchronization between data sources and repositories was ensured via programmatic data loading

3. Uniqueness
Data follows the definition standards for data type,size and format

4. Accuracy
Values used for measurements appeared in acceptable ranges

5. Integrity
Validity of data across the database is ensured by table-keys relationships and unique constraints

6. Timeless
Repository data is available on demand and in real time via SQL querying

Data Platform

Data Preparation Steps:

1. Downloading the Data - Using API for downloading the datasets by pulling it using Python
2. Filtering Data for Date Range Required - Filtered the datasets using R and Python
3. Removing Unnecessary Columns - Used OpenRefine, R and Python for deleting the removing unnecessary columns and removing the NA values.
4. Compile Similar Datasets - Used SQL for combining the datasets

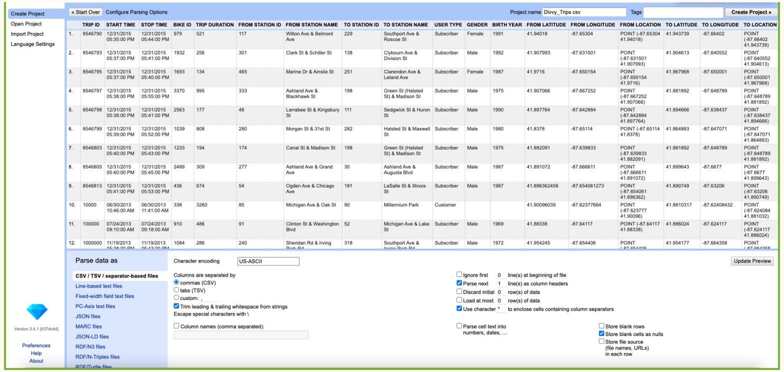
Platform Considerations:

- Query the API and download the datasets: Python
- Data cleaning : OpenRefine, R and Python
- Build the Relational Database and EER Diagram: MySQL
- Data visualizations: Tableau
- Remote team-work : GCP

Cleanup

Given the various datasets and skill level on the team, we used three different methods to clean and sort data

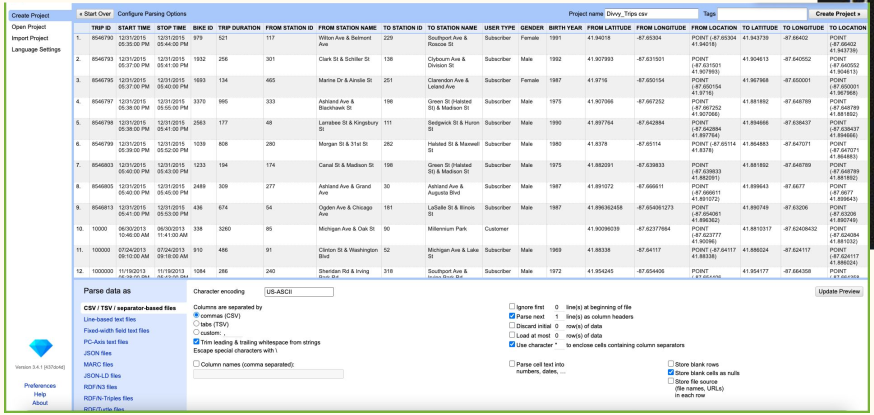
- OpenRefine
- Python
- R



RStudio and OpenRefine-Cleanup

- ❖ Examining our data
- ❖ Removing whitespace
- ❖ Changing the case
- ❖ Cleaning through cluster and edit

```r
install.packages("RSocrata")
LRidership <- read.socrata("https://data.cityofchicago.org/resource/5neh-572f.json")
LRidership <- LRidership[c(1:3,5)]
LRidership$station_id <- as.integer(LRidership$station_id)
LRidership$rides<- as.integer(LRidership$rides)
LRidership$date <- as.Date(LRidership$date)
head(LRidership)
final.LRidership <- subset(LRidership, LRidership$date > '2018-01-01')
final.LRidership <- final.LRidership[order(final.LRidership$date),]

write.csv(final,'Lridership2.csv',row.names=FALSE)

head(final.LRidership)
unique(final.LRidership$stationname)


L_STationNAmes <- read_csv("L-STationNAmes.csv")
L_STationNAmes$CommunityName <-tolower(L_STationNAmes$CommunityName)
df<- merge(L_STationNAmes,Community_Areas, by = 'CommunityName', all.x = TRUE)
head(df)
df$stationname <- tolower(df$stationname)
Lridership$stationname <- tolower(Lridership$stationname)

Lridership <- read_csv("Lridership.csv")
final<- merge(Lridership,df, by = 'stationname', all.x = TRUE)

taxi<- rbind(Taxi_Data_Set_2018,Taxi_Data_Set_2019,Taxi_Data_Set_2020)
```



Python - Cleanup

```
In [15]: results_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 23 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   trip_id                     2000 non-null   object
 1   taxi_id                     2000 non-null   object
 2   trip_start_timestamp        2000 non-null   object
 3   trip_end_timestamp          2000 non-null   object
 4   trip_seconds                2000 non-null   object
 5   trip_miles                  2000 non-null   object
 6   pickup_community_area       1876 non-null   object
 7   dropoff_community_area      1854 non-null   object
 8   fare                        2000 non-null   object
 9   tips                        2000 non-null   object
 10  extras                      2000 non-null   object
 11  trip_total                  2000 non-null   object
 12  payment_type                2000 non-null   object
 13  company                     2000 non-null   object
 14  pickup_centroid_latitude    1876 non-null   object
 15  pickup_centroid_longitude   1876 non-null   object
 16  pickup_centroid_location    1876 non-null   object
 17  dropoff_centroid_latitude   1854 non-null   object
 18  dropoff_centroid_longitude  1854 non-null   object
 19  dropoff_centroid_location   1854 non-null   object
 20  pickup_census_tract         1200 non-null   object
 21  dropoff_census_tract        1197 non-null   object
 22  tolls                       1737 non-null   object
dtypes: object(23)
memory usage: 359.5+ KB
```

```
In [16]: results_df.isnull().sum()

Out[16]: trip_id                         0
taxi_id                         0
trip_start_timestamp            0
trip_end_timestamp              0
trip_seconds                    0
trip_miles                      0
pickup_community_area         124
dropoff_community_area        146
fare                            0
tips                            0
extras                          0
trip_total                      0
payment_type                    0
company                         0
pickup_centroid_latitude      124
pickup_centroid_longitude     124
pickup_centroid_location      124
dropoff_centroid_latitude     146
dropoff_centroid_longitude    146
dropoff_centroid_location     146
pickup_census_tract           800
dropoff_census_tract          803
tolls                         263
dtype: int64
```

```
In [17]: #Checking the total and percent of the missing values
total = results_df.isnull().sum().sort_values(ascending=False)
percent = (results_df.isnull().sum()/results_df.isnull().count()).sort_values(ascending
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
```

Out[17]:

	Total	Percent
dropoff_census_tract	803	0.4015
pickup_census_tract	800	0.4000
tolls	263	0.1315
dropoff_centroid_location	146	0.0730
dropoff_centroid_longitude	146	0.0730
dropoff_centroid_latitude	146	0.0730
dropoff_community_area	146	0.0730
pickup_community_area	124	0.0620
pickup_centroid_location	124	0.0620
pickup_centroid_longitude	124	0.0620
pickup_centroid_latitude	124	0.0620
taxi_id	0	0.0000
trip_start_timestamp	0	0.0000
trip_end_timestamp	0	0.0000
trip_seconds	0	0.0000
trip_miles	0	0.0000
trip_total	0	0.0000
fare	0	0.0000
tips	0	0.0000
extras	0	0.0000

```
In [7]: results_df=results_df.dropna()

In [8]: results_df=results_df.drop(['taxi_id','payment_type','pickup_centroid_latitude','pickup

In [9]: results_df
```

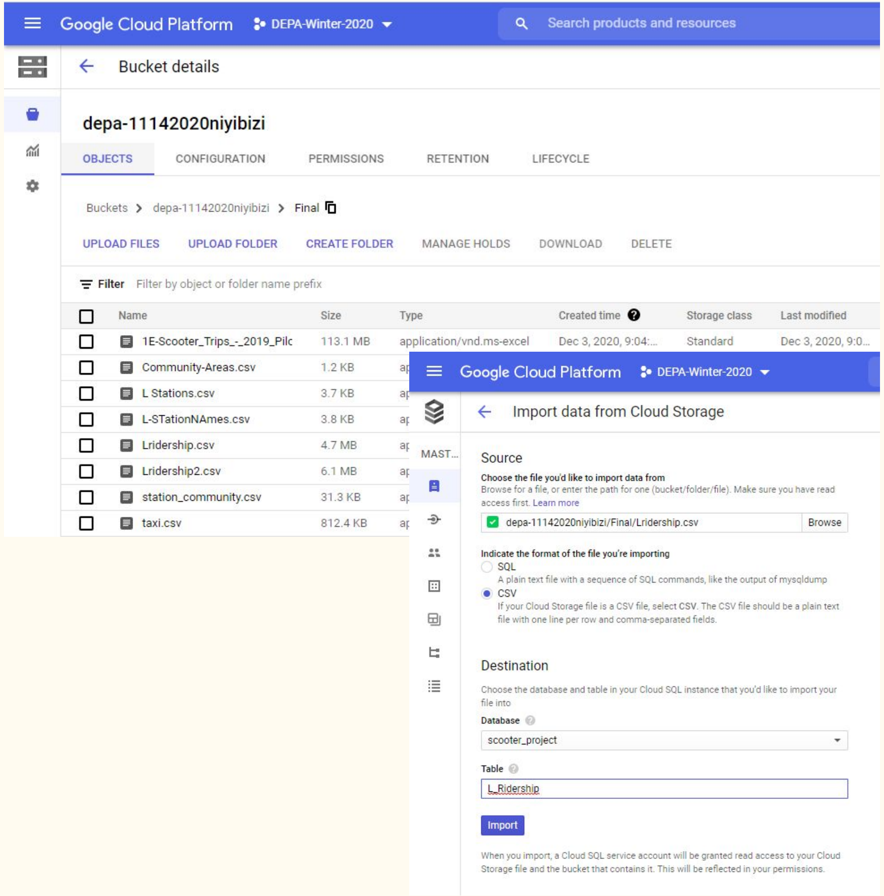
FOUR

Platform

Google Cloud Platform

Using the storage and and SQL modules on GCP we were able to:

- Store our datasets on the Google cloud which alleviates the need for storage space on our local machines and allowed for group members to access datasets
- Push said datasets from storage directly into a MySQL server on the cloud in an effective manner given the size of our large datasets



Data Consideration for RDBMS

Data Integrity

1. Establish Unique Primary Key for each Entity/Table
2. Define NULL explicitly for columns that are undefined
3. Establish foreign key relationship and constraints (NotNull, Unique)
4. Define Default values for missing attributes whenever applicable

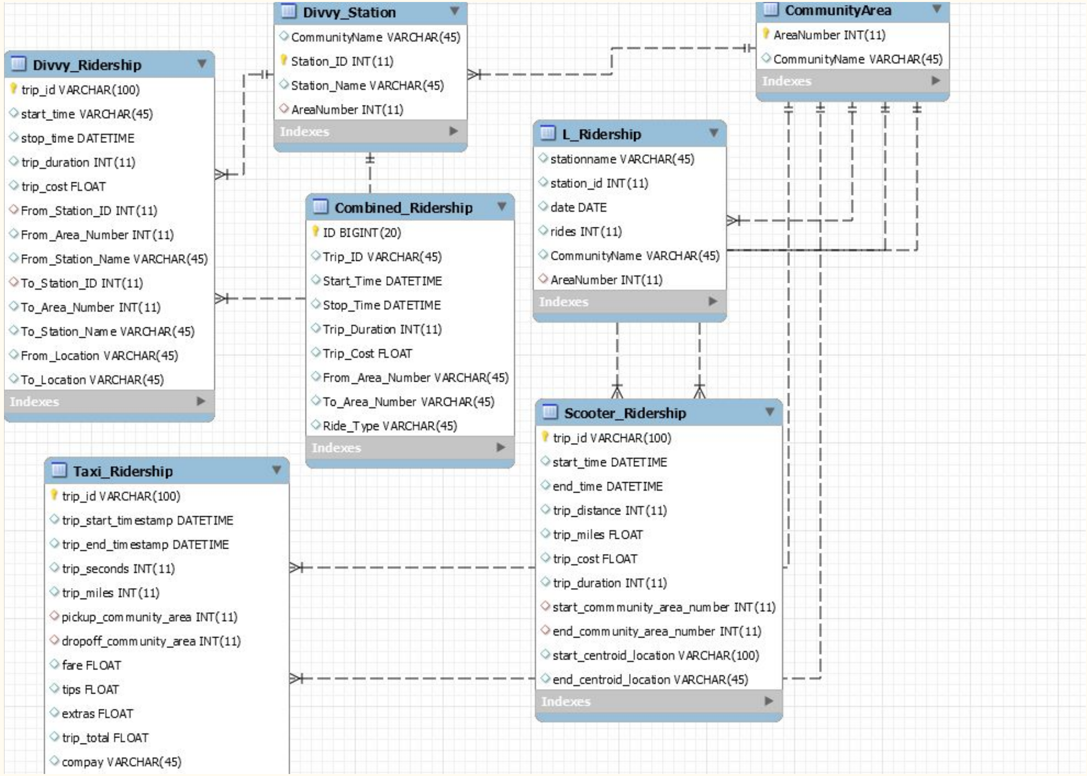
Data Types / Indexing

1. Choose Integer data types for Primary key and numbers
2. Define Data attributes (Ex. Date, TimeStamp, etc)
3. Follow standard naming conventions for attributes
4. Create Index for frequently queried columns like date
5. Partition the tables with data attributes by date

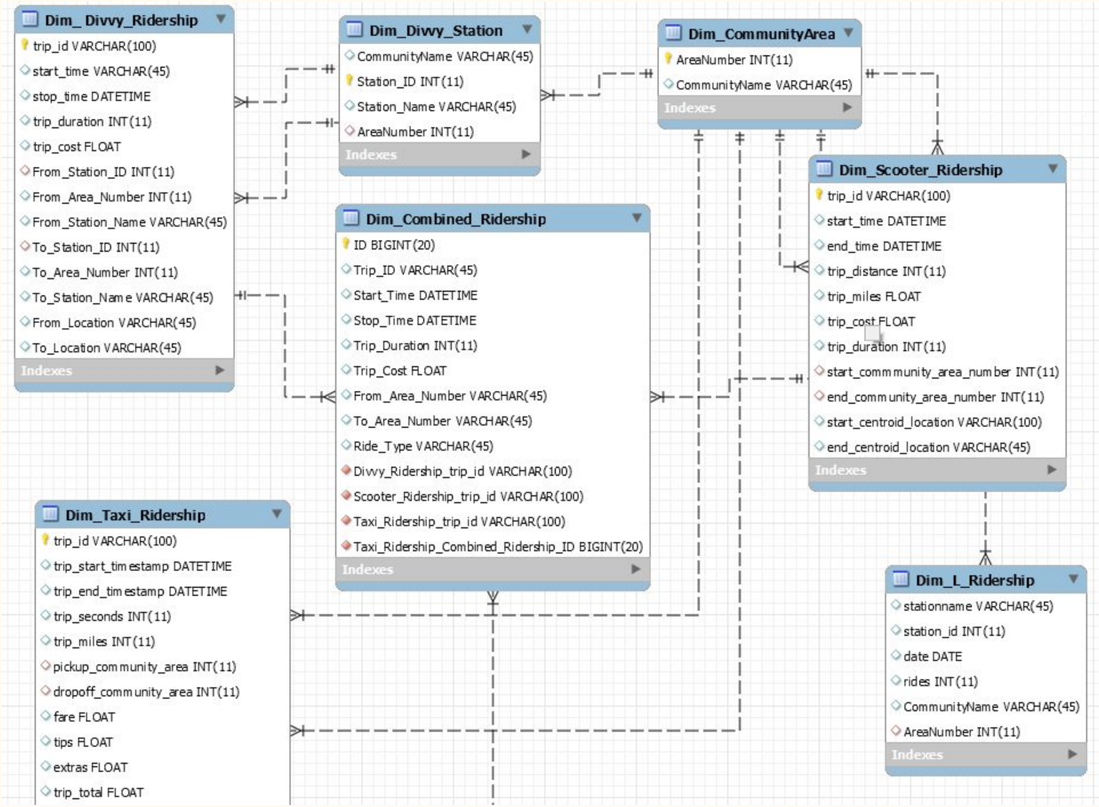
Database Modeling

1. OLTP - Normalized Physical Entity-Relationship Model
2. OLAP - Multidimensional Snowflake Model

ERR Model


Divvy_Ridership
trip_id VARCHAR(100)
start_time VARCHAR(45)
stop_time DATETIME
trip_duration INT(11)
trip_cost FLOAT
From_Station_ID INT(11)
From_Area_Number INT(11)
From_Station_Name VARCHAR(45)
To_Station_ID INT(11)
To_Area_Number INT(11)
To_Station_Name VARCHAR(45)
From_Location VARCHAR(45)
To_Location VARCHAR(45)
Indexes
Divvy_Station
CommunityName VARCHAR(45)
Station_ID INT(11)
Station_Name VARCHAR(45)
AreaNumber INT(11)
Indexes
CommunityArea
AreaNumber INT(11)
CommunityName VARCHAR(45)
Indexes
L_Ridership
stationname VARCHAR(45)
station_id INT(11)
date DATE
rides INT(11)
CommunityName VARCHAR(45)
AreaNumber INT(11)
Indexes
Combined_Ridership
ID BIGINT(20)
Trip_ID VARCHAR(45)
Start_Time DATETIME
Stop_Time DATETIME
Trip_Duration INT(11)
Trip_Cost FLOAT
From_Area_Number VARCHAR(45)
To_Area_Number VARCHAR(45)
Ride_Type VARCHAR(45)
Indexes
Scooter_Ridership
trip_id VARCHAR(100)
start_time DATETIME
end_time DATETIME
trip_distance INT(11)
trip_miles FLOAT
trip_cost FLOAT
trip_duration INT(11)
start_community_area_number INT(11)
end_community_area_number INT(11)
start_centroid_location VARCHAR(100)
end_centroid_location VARCHAR(45)
Indexes
Taxi_Ridership
trip_id VARCHAR(100)
trip_start_timestamp DATETIME
trip_end_timestamp DATETIME
trip_seconds INT(11)
trip_miles INT(11)
pickup_community_area INT(11)
dropoff_community_area INT(11)
fare FLOAT
tips FLOAT
extras FLOAT
trip_total FLOAT
compay VARCHAR(45)

Dimension Model



Data Mapping using SQL

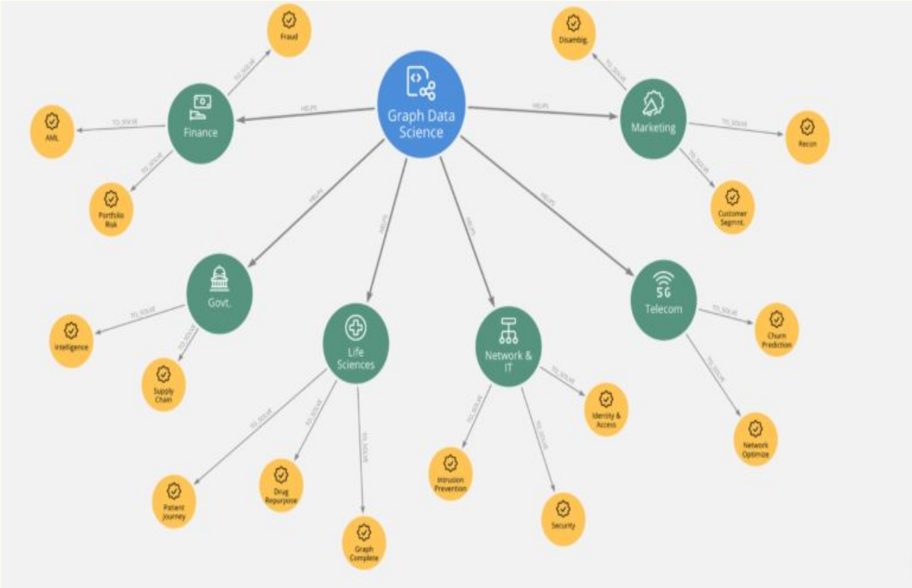
- Leveraging GCP we were able to push our data sets into the tables we had built out in MySQL
- Additionally, using SQL, we were able to combine the various data sets to create our final table to read from
- From there we were able to push the tables into Tableau were we could create dashboards and draw insights

```
drop table my_tmp_table;
CREATE TEMPORARY TABLE my_tmp_table
 (Ridership_ID BIGINT NOT NULL AUTO_INCREMENT, PRIMARY KEY my_pkey (Ridership_ID), INDEX my_unique_index_name (Ridership_ID))
 select * from(
select trip_id, start_time,stop_time, trip_duration,Trip_Cost, From_Area_Number,To_Area_Number, ('Divvy') as Ride_type from Divvy_Ridership d
union all
select trip_id, start_time, end_time, trip_duration, Trip_Cost, start_commmunity_area_number, end_community_area_number, ('Scooter') as Ride_t
union all
select trip_id, trip_start_timestamp, trip_end_timestamp, trip_seconds, trip_total, pickup_community_area, dropoff_community_area,('Taxi') as
) as ride_table;

insert into Final select * from my_tmp_table;
select * from Final limit 5;
```

```
-- -----------------------------------------------------

DROP TABLE IF EXISTS `scooter_project`.`Divvy_Ridership` ;

CREATE TABLE IF NOT EXISTS `scooter_project`.`Divvy_Ridership` (
  `trip_id` VARCHAR(100) NOT NULL,
  `start_time` DATETIME NULL DEFAULT NULL,
  `stop_time` DATETIME NULL DEFAULT NULL,
  `trip_duration` INT(11) NULL DEFAULT NULL,
  `From_Station_ID` INT(11) NULL DEFAULT NULL,
  `From_Station_Name` VARCHAR(45) NULL DEFAULT NULL,
  `To_Station_ID` INT(11) NULL DEFAULT NULL,
  `To_Station_Name` VARCHAR(45) NULL DEFAULT NULL,
  `From_Location` VARCHAR(45) NULL DEFAULT NULL,
  `To_Location` VARCHAR(45) NULL DEFAULT NULL,
  `Trip_Cost` FLOAT NULL DEFAULT NULL,
  PRIMARY KEY (`trip_id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf16
INSERT_METHOD = LAST;



-- -----------------------------------------------------
-- Table `scooter_project`.`L_Ridership`
```

Neo4j

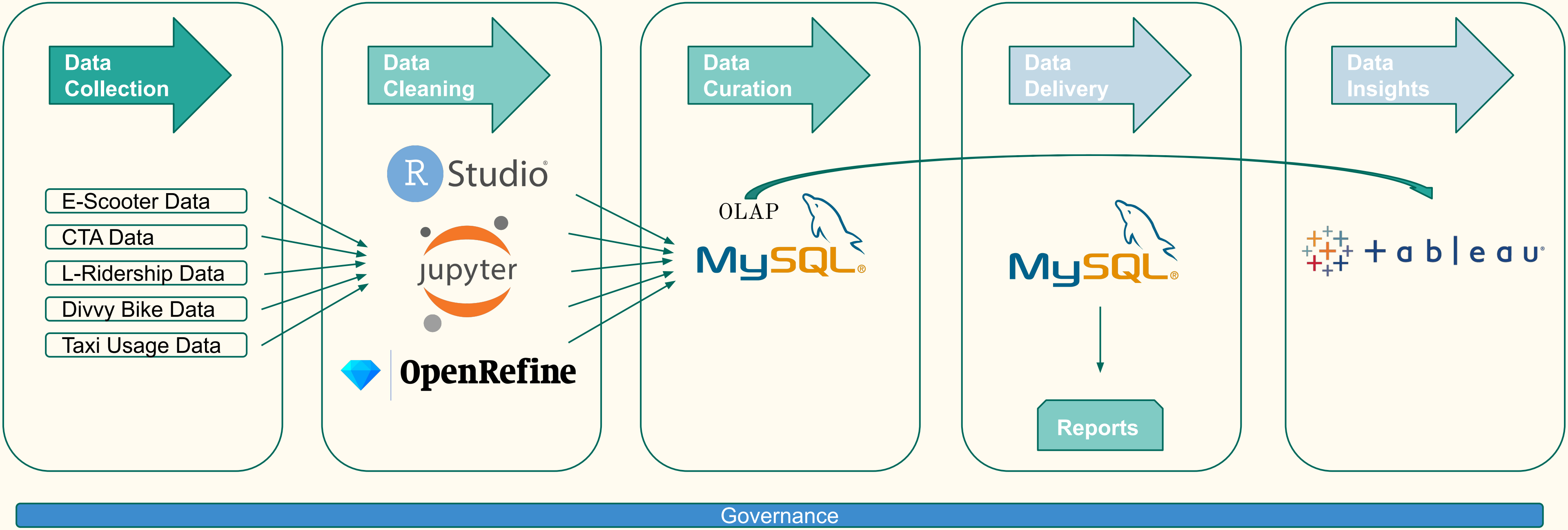
- For data visualization of transportation in Chicago, Neo4j is a graph database management system with native graph storage and processing
- Schema free nature help recognize new relationships
- Data access: beneficial as time-based data continues to grow (naturally additive)
- Nodes: Rider type - To Community ID and From Community ID
- Use labels related to:distance and Rider Type Relationship: distance and cost
- Identify clusters or growing clusters over time



MongoDB

- From the website *the city of Chicago,* collected JSON-like datasets. MongoDB provides commercial support for NoSQL database that stores data in JSON-like documents with flexible schemas
- Import our datasets, combine them for the future manipulation
- Disjoint data would benefit from the minimal constraints of simplified queries
- More flexible data store for new data points

Data Platform



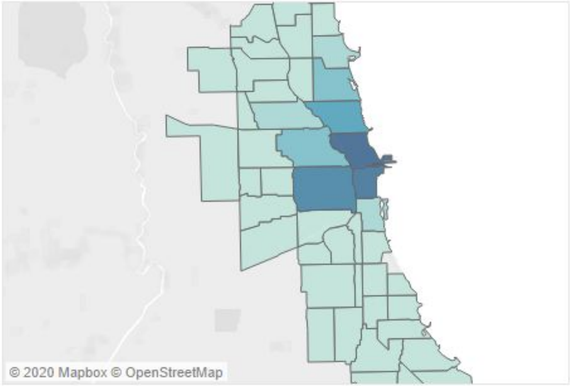
FIVE

Insights

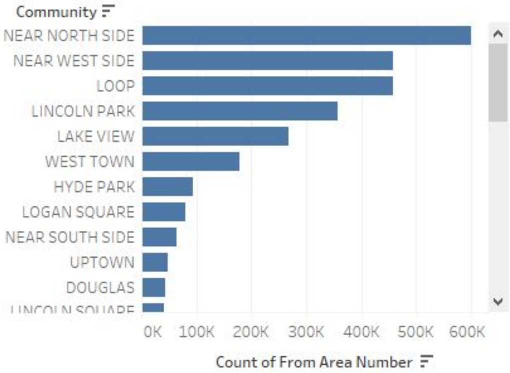
—

Divvy Dashboard

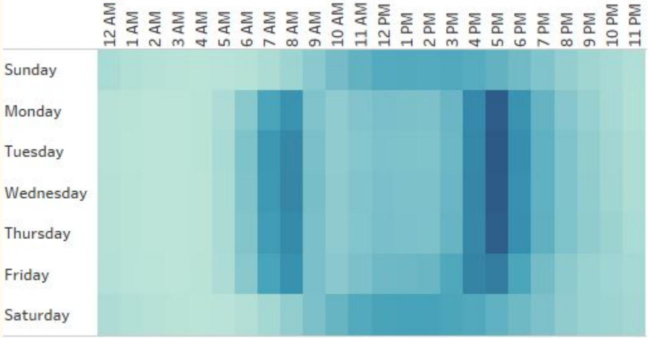
Top Community Areas



Top Community - Graph



Busiest Times



Overall Ridership

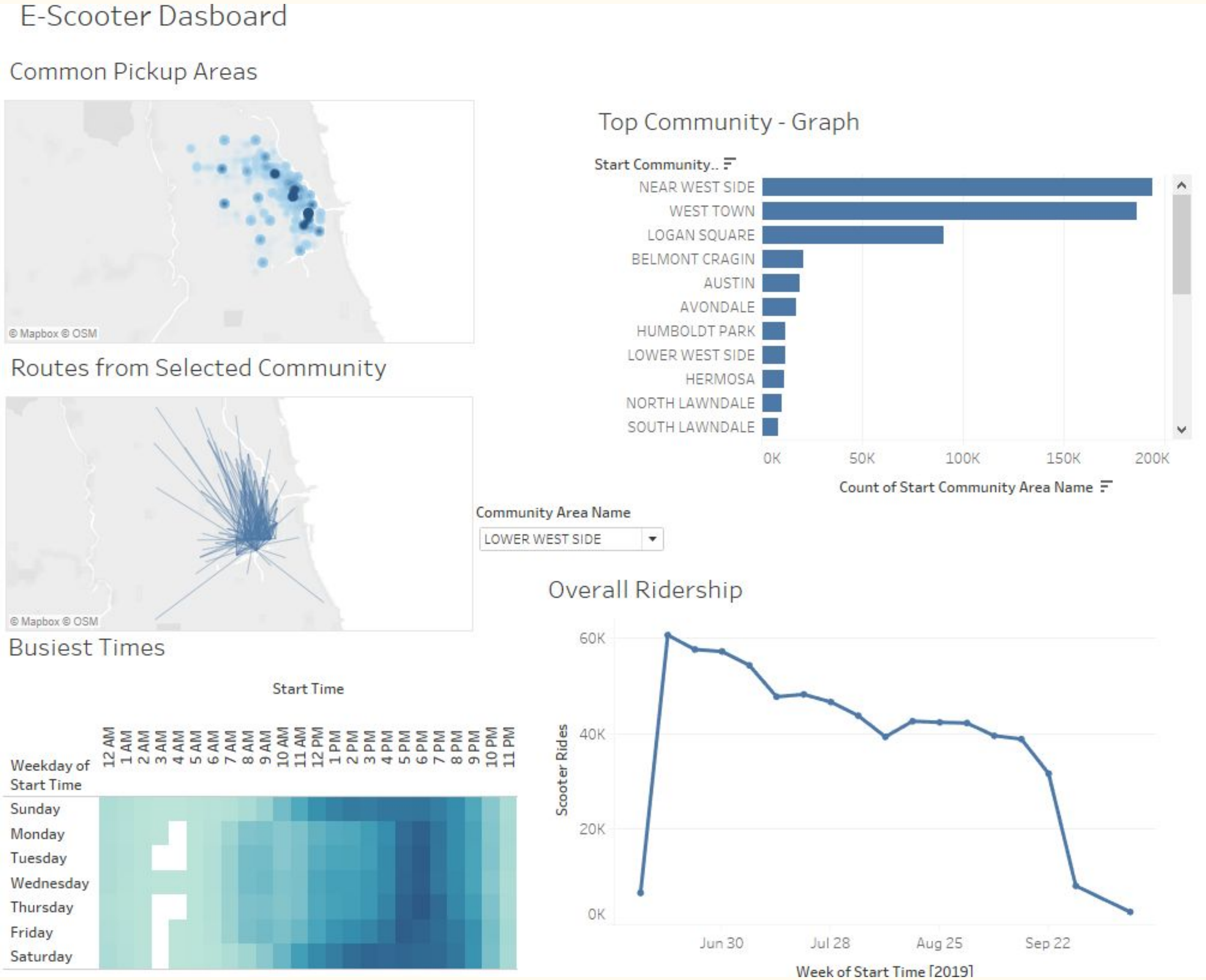
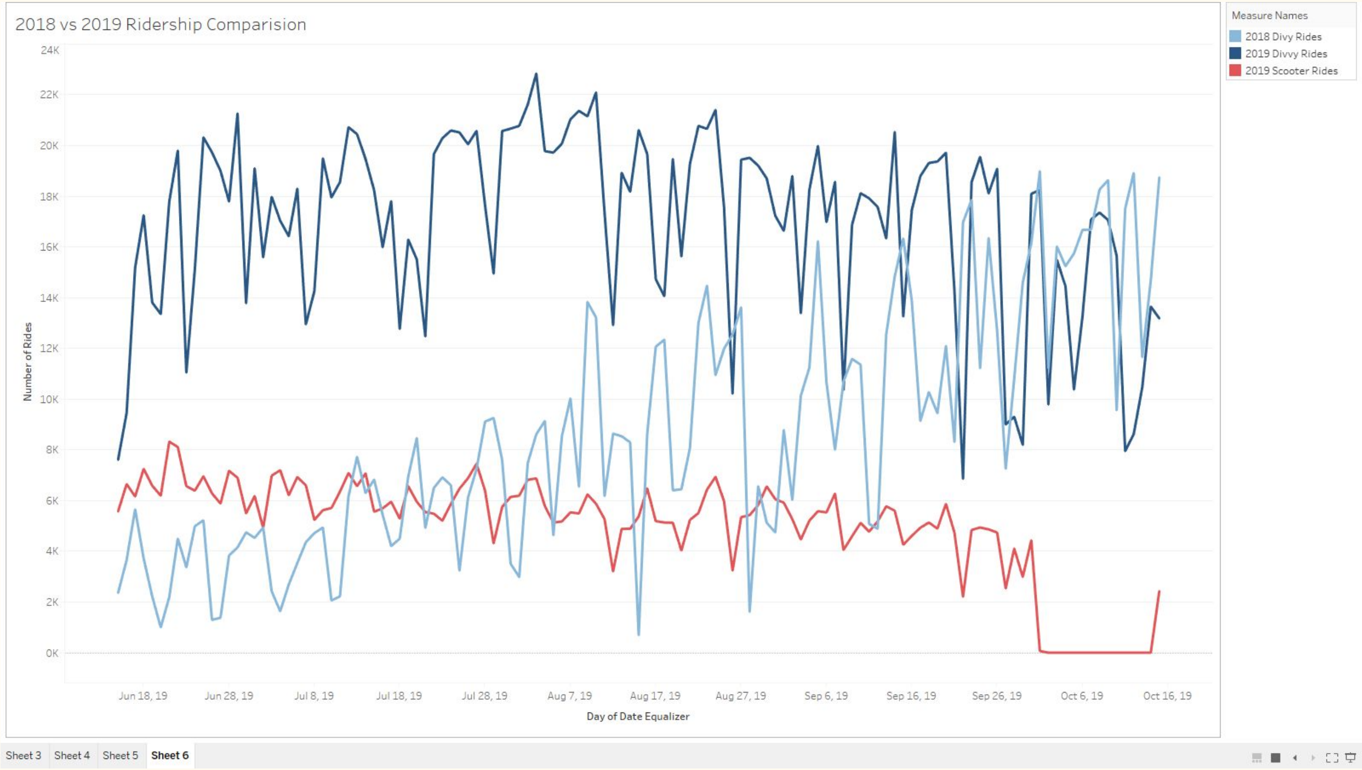
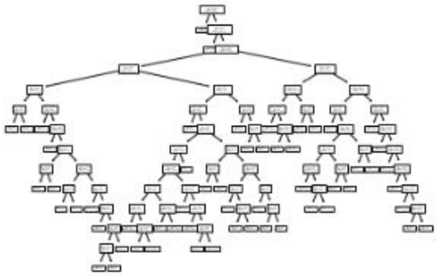

E-Scooter Dasboard
Common Pickup Areas
© Mapbox © OSM
Routes from Selected Community
© Mapbox © OSM
Busiest Times
Start Time
Weekday of Start Time
12 AM
1 AM
2 AM
3 AM
4 AM
5 AM
6 AM
7 AM
8 AM
9 AM
10 AM
11 AM
12 PM
1 PM
2 PM
3 PM
4 PM
5 PM
6 PM
7 PM
8 PM
9 PM
10 PM
11 PM
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Top Community - Graph
Start Community..
NEAR WEST SIDE
WEST TOWN
LOGAN SQUARE
BELMONT CRAGIN
AUSTIN
AVONDALE
HUMBOLDT PARK
LOWER WEST SIDE
HERMOSA
NORTH LAWNDALE
SOUTH LAWNDALE
0K
50K
100K
150K
200K
Count of Start Community Area Name
Community Area Name
LOWER WEST SIDE
Overall Ridership
60K
40K
20K
0K
Scooter Rides
Jun 30
Jul 28
Aug 25
Sep 22
Week of Start Time [2019]

Tableau Insights



Using Random Forest in Python

```
In [39]: from sklearn.tree import DecisionTreeClassifier
         from sklearn.model_selection import train_test_split
         from sklearn import metrics
         X = new_df.drop(columns=['Ride_type_y'])
         y = new_df['Ride_type_y']
         X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2)

         model = DecisionTreeClassifier()
         model.fit(X_train,y_train)

Out[39]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                                max_depth=None, max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort='deprecated',
                                random_state=None, splitter='best')

In [47]: from matplotlib import pyplot as plt
         from sklearn import tree
         fig = model.fit(X_train,y_train)

In [48]: tree.plot_tree(fig)
         plt.show()
```



SIX

Conclusion

Conclusion

Analyze the E-scooter pilot program in addition to other methods of transportation and their impacts to better prepare governmental agencies when planning transportation infrastructure for their communities.

Key Stats

34,405
E-Scooter Rides

June 15th to
Oct 15th 2019
Pilot Duration

Findings and Recommendations

Increase self-operated short-form transportation method

Transportation Regulation Changes

Limitations

Excluded certain vehicular transportation methods

Excluded reasons for transportation

Team

Akhil Ranjan

Role: Data Scientist, TransAnalytics

Education:
UChicago MScA
VIT University - B.Tech

Olga Niyibizi

Role: Data Scientist, TransAnalytics

Education:
UChicago MScA
Saint Mary's College

Sahil Sachdev

Role: Data Scientist, TransAnalytics

Education:
UChicago MScA
USC BA

Jingyu Zhang

Role: Data Scientist, TransAnalytics

Education:
UChicago MScA
UIUC BS

Thank You!

References

https://data.cityofchicago.org/

https://pubmed.ncbi.nlm.nih.gov/2761343

https://multimedia.journalism.berkeley.edu/tutorials/openrefine/

https://www.mongodb.com/what-is-mongodb

https://en.wikipedia.org/wiki/Neo4j