NYC Taxi Data Warehouse

CSCI 5707 Project Report

December 12th, 2019

—

Group 6

Krishna Shravya Gade - 5592616

Mourya Karan Reddy Baddam - 5564234

Rutvij Bora - 5594898

Ranjabati Sen - 5586108

Table of Contents

1. Introduction

When the ride hailing apps like Uber and Lyft started their business in New York City, they faced a cutthroat competition from the already existing Taxi network which was well established. The taxi licensing system is based on taxi medallion which is a transferable permit for the driver. A taxi medallion in NYC was valued over $1,000,000 in 2013 [2]. Setting up a new medallion not only costed a fortune but also meant dealing with the underlying power dynamics with the competitors. Then came the ride hailing apps like Uber who with their business models single handedly transformed the structure of the business. As of June 2019, a taxi medallion in NYC costs less than $138,000 if its lucky enough to get bidders [2]. These companies had the power of data on their side and using the user statistics, individual trip data and driver statistics, they were able to deliver exactly what the industry needed [1]. Our idea is to use that available NYC TLC data to deliver some of these benefits to the NYC taxi network.

2. Objective

Our goal in this project is to use the existing NYC TLC data which is a month wise aggregate data for Yellow, Green and HFV taxis in NYC that has trip start and end times and locations along with fare amounts and trips to give analytics and answer some critical business questions to improve the business model.

Having said that, we have also focused on a specific business question which might not be relevant to improving taxi network performance but beneficial for setting up new businesses in NYC to demonstrate that this analytics tool is multifaceted and can be used to answer a multitude of business questions.

The business questions we are looking to answer with respect to the driver statistics are something like, “Where can I go on a Sunday afternoon to get more ride pickups?” Or “I want to be home by 8PM today, which location can I go for pickups so that there is a higher chance of the drop being close to my home?” The benefit of these kind of questions is that it not only makes the driver’s life easier, it will increase the number of trips per vehicle giving a higher vehicle utilization. The third kind of business question is, “I want to set up a coffee house in NYC, which locations are my best bet?” To answer this we would look for locations that have more drop offs during morning hours when people prefer to have coffee finding the busy areas at this time.

3. Datasets

NYC Taxi and Limousine Commission Trip Record Data

Data Source: The NYC Taxi and Limousine Commission website [3]

Description: Dataset provided by the NYC Taxi and Limousine Commission (TLC) containing extensive information about the New York City taxi trips since 2009. The data is available on a monthly basis and its format changed mid 2016, with the inclusion of pickup and drop off zone IDs. These zone ids can be used to link this dataset to NYC Taxi Zone Reference Data for retrieving geospatial coordinates. Since these coordinates are required to answer our business questions regarding this dataset, we only used data after this change. The data belonging to yellow and green taxi data have many overlapping fields but with a few changes in their schema.

Format: collection of csv files

Scale:

- File size: 58GB (total)
- Number of rows: 2.2 Billion
- Number of columns: 18 for yellow taxi and 20 for green taxi

Field	Type	Description	Sample value
pickup_datetime	datetime	The date and time when the meter was engaged.	2019-01-01 00:46:40
PULocationID	string	TLC Taxi Zone in which the taxi meter was engaged. Used to link with taxi zone data (explained below)	151
dropoff_datetime	datetime	The date and time when the meter was disengaged.	151
DOLocationID	string	TLC Taxi Zone in which the taxi meter was disengaged	239
tip_amount	double	Tip amount	1.5
total_amount	double	The total amount charged to passengers	9.95
+11 fields related to charges like tax, vendors and passengers			

NYC Taxi Zone Reference Data

Data Source: NYC Open Data [3]

Description: Dataset provided by the NYC Taxi and Limousine Commission to give spatial context to the 263 ‘taxi zones’ referenced in the taxi trip record data. It contains information regarding the encompassing borough (town or district) name and geospatial coordinates.

Format: csv

Scale:

- File size: 3.8 MB
- Number of rows: 263
- Number of columns: 5

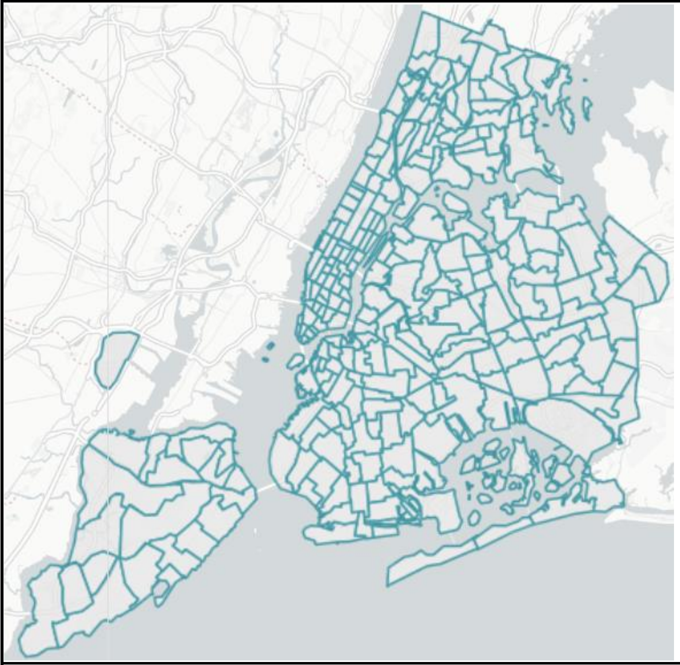


Figure 1: Map of Taxi zones across NYC

Field	Type	Description	Sample value
OBJECTID	Integer	Equivalent to LocationID	239
the_geom	MultiPolygon	Spatial data polygon defining zone borders	MULTIPOLYGON (((-74.18445299999996 40.694995999999904, ..)
zone	String	Zone name	Auburndale
borough	String	Borough name	Queens

4. Technologies

Python 3.7 Pandas and NumPy Framework

We chose pandas framework to download data and perform data cleaning. There were many null and negative fare values, 0 distance trips and trips that had 0 fare but positive distance. These rows were eliminated by us to avoid aberrations in the means and standard deviation values which were used in our analysis. The pandas version used was 0.25.3 along with NumPy version 1.17.4 for conditional operations.

MySQL 8.0.18

We chose MySQL 8.0 as our database system for the SQL queries. The reasons behind using MySQL was that it was easy to install and did not require high end specification apart from being free of cost unlike Oracle. The functionalities were straightforward and we needed the simplest implementation of purely relational database unlike PostgreSQL. But the disadvantage was that MySQL does not support materialized views which could have improved the space utilization and overall long term performance of our application.

Neo4J 3.5.8

- Neo4J is a graph database with native graph storage and processing.
- It's property graph model and cypher query language make it easy to understand.
- We wanted to try out loading the trips as relationships between each location nodes.

OrientDB 3.0.24

OrientDB is a Multi-Model database with huge storage capacity and following capabilities:

- There can be more than 1000 databases open at the same time.
- Each Database can have upto 32767 clusters.
- Each cluster can hold upto 10 trillion records with upto size of 2 GB per record.
- Indexing on fields is supported with upto 2 Billion Indexes per database and unlimited indices per class.
- These features matched our requirement of storing and querying millions of records.
- OrientDB provides a Java API to connect to a Java Application

5. Summarization

ER-Diagram

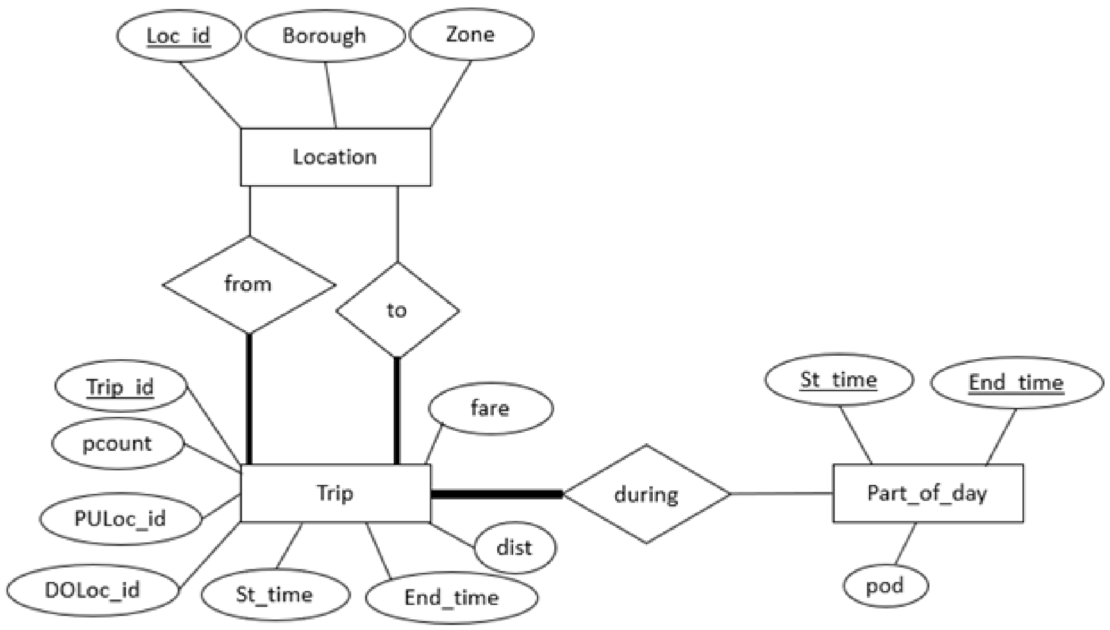


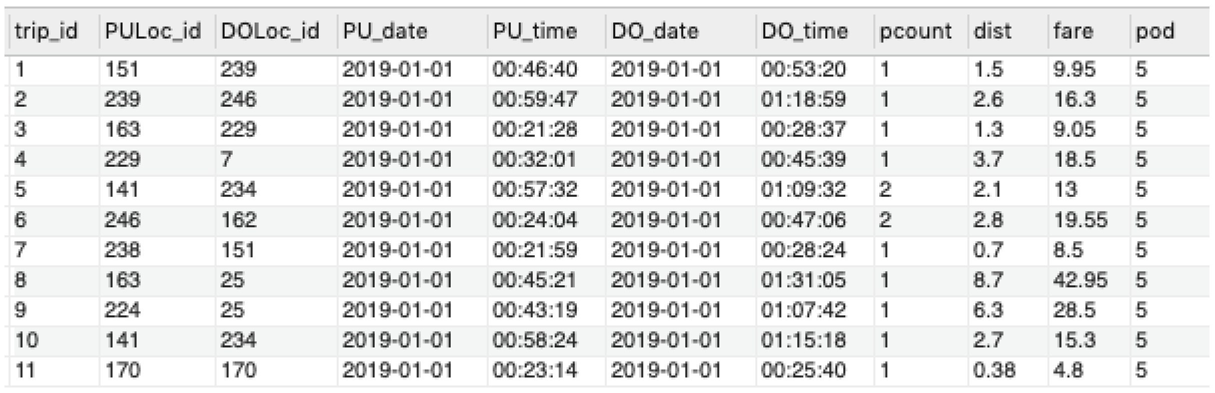
Figure 1: Entity Relationship diagram for NYC Taxi dataset

The ER-diagram for the NYC taxi database is as shown above. It has three Entities – Trip, Part_of_day and Location. They are related by three relationships – Trip *during* Part_of_day, Trip from Location and Trip *to* Location. There are completeness constraints on Trip Entity in all the three relationships indicating that each trip must have a part of day (based on St_time) and each trip must have a *from* and *to* location.

Our table of interest for warehousing is the Trip Entity table (which is named as *jan_2019* in the *nyctaxi* schema), a sample of which is shown below. The number of Yellow taxi trips for the month of January 2019 is 7,667,794. But as a few of them had spurious values like zero fare for non-zero distance and vice versa, we had to remove such trips. We cleaned the data using python pandas and NumPy.

The part of the day is obtained by dividing the day into 5 parts based on peak hours (6 am – 10 am & 4 pm – 8pm) and non-peak hours of New York City as defined by MTA (Metropolitan Transportation Authority).

Start time	6:00:01 - 10:00:00	16:00:01 - 20:00:00	10:00:01 - 16:00:00	20:00:01 - 00:00:00	00:00:01 - 06:00:00
POD	1	2	3	4	5

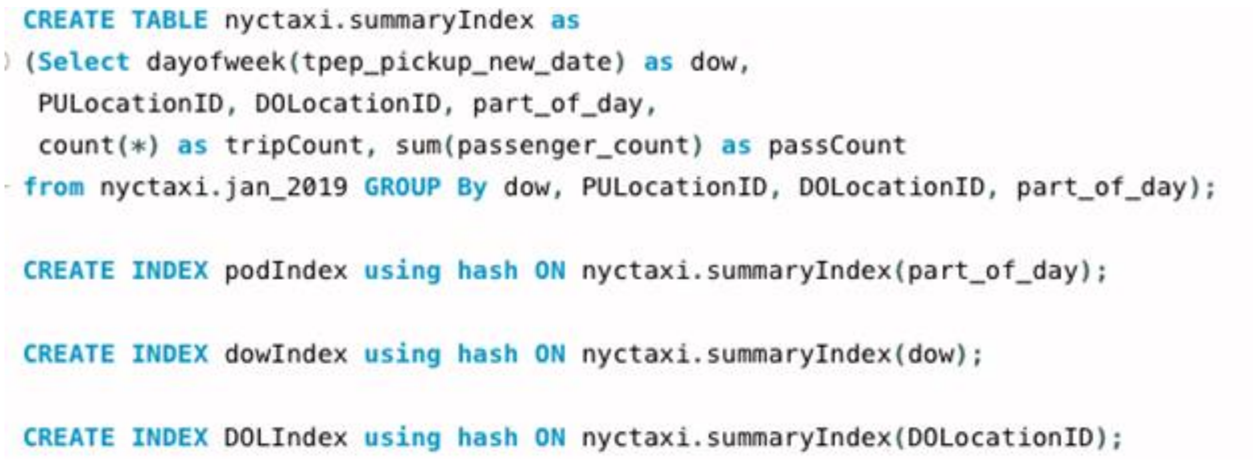


trip_id	PULoc_id	DOLoc_id	PU_date	PU_time	DO_date	DO_time	pcount	dist	fare	pod
1	151	239	2019-01-01	00:46:40	2019-01-01	00:53:20	1	1.5	9.95	5
2	239	246	2019-01-01	00:59:47	2019-01-01	01:18:59	1	2.6	16.3	5
3	163	229	2019-01-01	00:21:28	2019-01-01	00:28:37	1	1.3	9.05	5
4	229	7	2019-01-01	00:32:01	2019-01-01	00:45:39	1	3.7	18.5	5
5	141	234	2019-01-01	00:57:32	2019-01-01	01:09:32	2	2.1	13	5
6	246	162	2019-01-01	00:24:04	2019-01-01	00:47:06	2	2.8	19.55	5
7	238	151	2019-01-01	00:21:59	2019-01-01	00:28:24	1	0.7	8.5	5
8	163	25	2019-01-01	00:45:21	2019-01-01	01:31:05	1	8.7	42.95	5
9	224	25	2019-01-01	00:43:19	2019-01-01	01:07:42	1	6.3	28.5	5
10	141	234	2019-01-01	00:58:24	2019-01-01	01:15:18	1	2.7	15.3	5
11	170	170	2019-01-01	00:23:14	2019-01-01	00:25:40	1	0.38	4.8	5

Figure 2: Trip table in NYC Taxi Database

Summarization

The Business questions that we are going to query about involves finding the number of trips and number of passengers that took the trips. We would have to use aggregate functions in SQL for every query and that is going to be computationally expensive. Summarizing the trips using aggregate functions beforehand helps make the computations faster. It also reduces the storage space required for the data. The jan_2019 table has been summarized using the query below.



```
CREATE TABLE nyctaxi.summaryIndex as
(Select dayofweek(tpep_pickup_new_date) as dow,
 PULocationID, DOLocationID, part_of_day,
 count(*) as tripCount, sum(passenger_count) as passCount
from nyctaxi.jan_2019 GROUP By dow, PULocationID, DOLocationID, part_of_day);

CREATE INDEX podIndex using hash ON nyctaxi.summaryIndex(part_of_day);

CREATE INDEX dowIndex using hash ON nyctaxi.summaryIndex(dow);

CREATE INDEX DOLIndex using hash ON nyctaxi.summaryIndex(DOLocationID);
```

The above SQL query creates a summary table in which the trips and the number of passengers who took the trips have been aggregated over day of the week, Pickup Location, Dropoff Location and part of the day. This summary table gives us the number of passengers and total number of trips from a given location to any other given location happened in a particular part of the day on a particular day of the week, for January 2019. The summary has also been indexed using Hash Index on part of the day, day of the week and Dropoff Location, as the queries use a point search on these attributes.

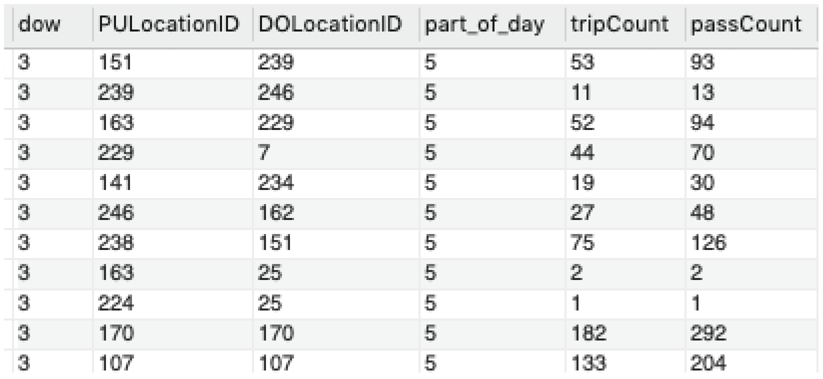
The summarization has led to a substantial reduction is the amount of data resulting in faster and efficient queries.

No. of rows in jan_2019 = 7,667,794
No. of rows in SummaryIndex = 316,817
Reduction Factor ≈ 24.

The DAYOFWEEK() function in MySQL returns an integer which represents the day in a week as per the below table.

Integer	Day of the week
1	Sunday
2	Monday
3	Tuesday
4	Wednesday
5	Thursday
6	Friday
7	Saturday

The figure shown below is representative of the summary table.



dow	PULocationID	DOLocationID	part_of_day	tripCount	passCount
3	151	239	5	53	93
3	239	246	5	11	13
3	163	229	5	52	94
3	229	7	5	44	70
3	141	234	5	19	30
3	246	162	5	27	48
3	238	151	5	75	126
3	163	25	5	2	2
3	224	25	5	1	1
3	170	170	5	182	292
3	107	107	5	133	204

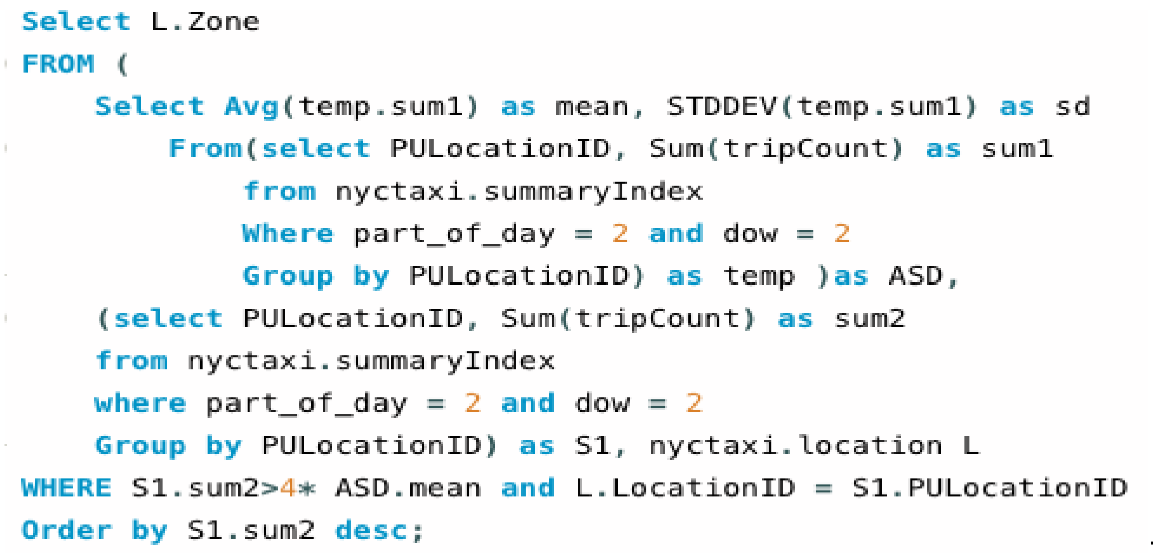
Figure 3: Summary table

6. Business Questions and Queries

Business question I: Pick-up Hotspots

We want to find the locations that have a high requirement of cabs at any given time on a particular day of the week.

We wrote the following SQL query to answer the business question:

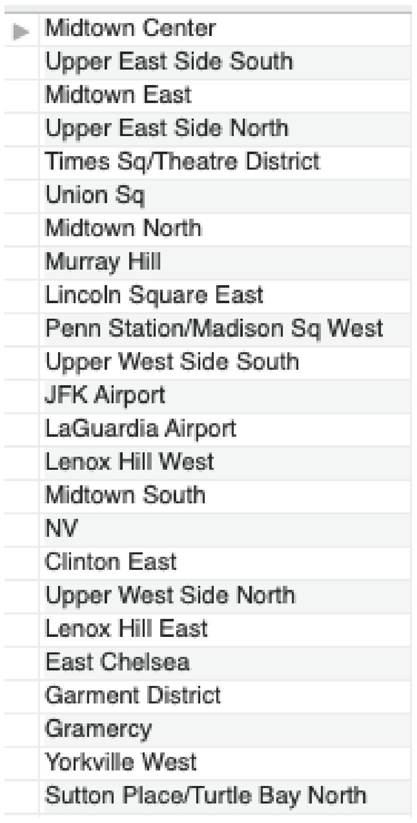


```
Select L.Zone
FROM (
    Select Avg(temp.sum1) as mean, STDDEV(temp.sum1) as sd
        From(select PULocationID, Sum(tripCount) as sum1
            from nyctaxi.summaryIndex
            Where part_of_day = 2 and dow = 2
            Group by PULocationID) as temp )as ASD,
    (select PULocationID, Sum(tripCount) as sum2
    from nyctaxi.summaryIndex
    where part_of_day = 2 and dow = 2
    Group by PULocationID) as S1, nyctaxi.location L
WHERE S1.sum2>4* ASD.mean and L.LocationID = S1.PULocationID
Order by S1.sum2 desc;
```

;

This query returns the pickup locations which have total number of trips more than 4 times the average number of pickups at given part of the day on a particular day of the week.

Result:

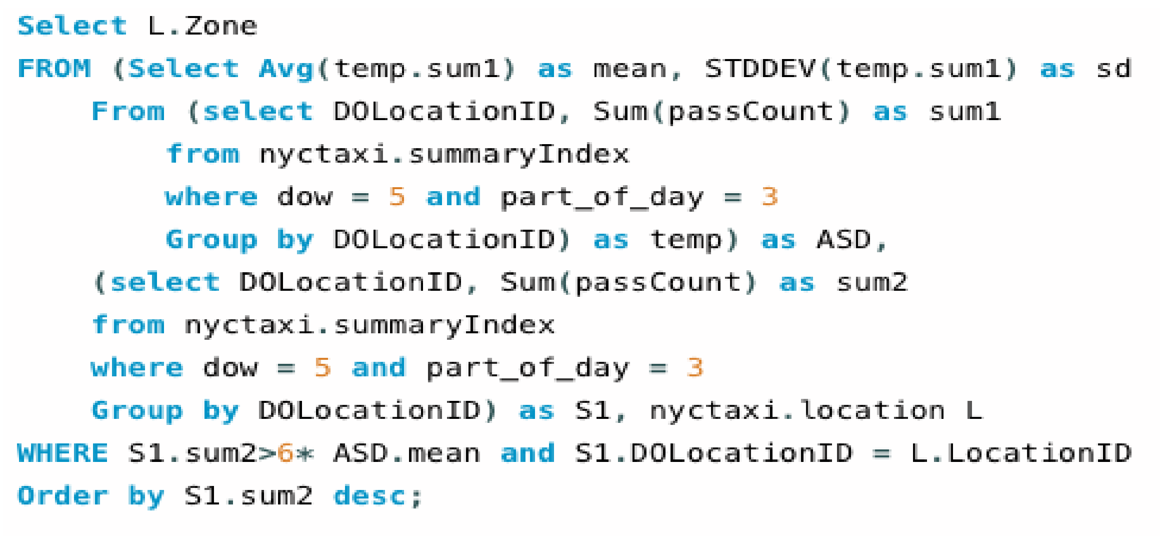


Midtown Center
Upper East Side South
Midtown East
Upper East Side North
Times Sq/Theatre District
Union Sq
Midtown North
Murray Hill
Lincoln Square East
Penn Station/Madison Sq West
Upper West Side South
JFK Airport
LaGuardia Airport
Lenox Hill West
Midtown South
NV
Clinton East
Upper West Side North
Lenox Hill East
East Chelsea
Garment District
Gramercy
Yorkville West
Sutton Place/Turtle Bay North

Business question II: Drop-off Hotspots

We want to find those locations that have the maximum number of visitors on a particular day of the week at a particular time. This information is useful in a scenario where supposing we want to open a restaurant at a place and we want to know the areas that are hotspots in a city so that the business can attract more customers.

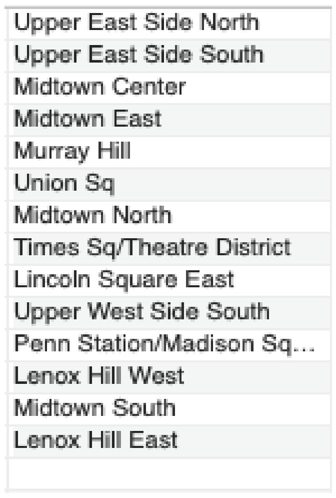
We wrote the following SQL query to answer this business question:



```
Select L.Zone
FROM (Select Avg(temp.sum1) as mean, STDDEV(temp.sum1) as sd
    From (select DOLocationID, Sum(passCount) as sum1
        from nyctaxi.summaryIndex
        where dow = 5 and part_of_day = 3
        Group by DOLocationID) as temp) as ASD,
    (select DOLocationID, Sum(passCount) as sum2
    from nyctaxi.summaryIndex
    where dow = 5 and part_of_day = 3
    Group by DOLocationID) as S1, nyctaxi.location L
WHERE S1.sum2>6* ASD.mean and S1.DOLocationID = L.LocationID
Order by S1.sum2 desc;
```

This query will return the list of drop locations with the number of visitors greater than a user specified threshold.

Result:

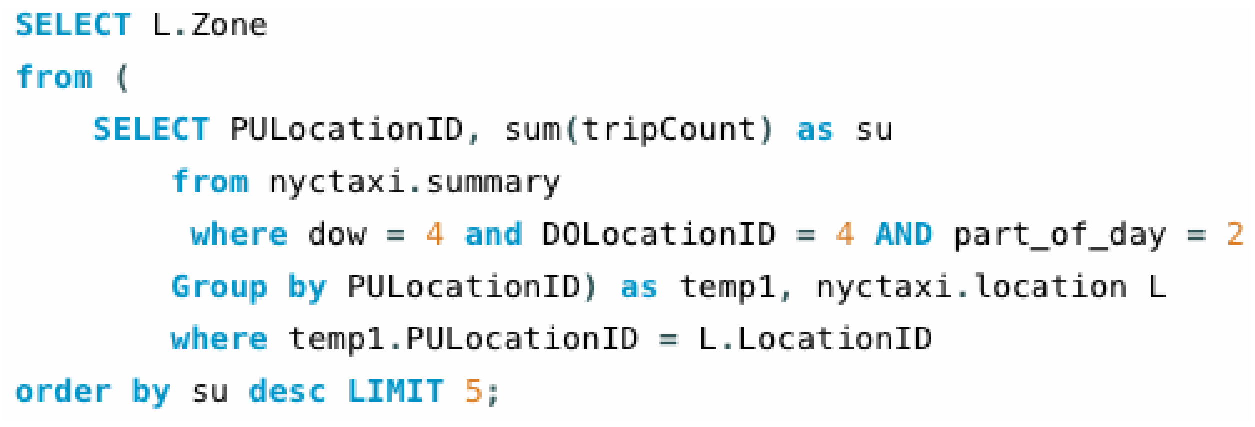


Upper East Side North
Upper East Side South
Midtown Center
Midtown East
Murray Hill
Union Sq
Midtown North
Times Sq/Theatre District
Lincoln Square East
Upper West Side South
Penn Station/Madison Sq…
Lenox Hill West
Midtown South
Lenox Hill East

Business question III: Driver preferred Drop-off

A taxi driver wants to reach a location X. He would like to take a trip with the drop off location as X. Which pickup location (near him) should he go to so that he has higher chances of getting a trip with its drop location as X?

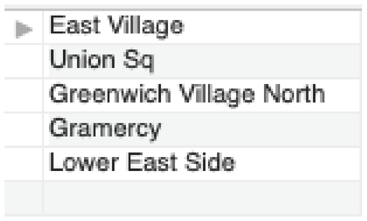
We wrote the following SQL query to answer this business question:



```
SELECT L.Zone
from (
    SELECT PULocationID, sum(tripCount) as su
        from nyctaxi.summary
         where dow = 4 and DOLocationID = 4 AND part_of_day = 2
        Group by PULocationID) as temp1, nyctaxi.location L
        where temp1.PULocationID = L.LocationID
order by su desc LIMIT 5;
```

This query will return the top 5 pickup locations with the highest probability of getting a trip to the driver's choice of destination on a particular day of the week at a particular time of the day.

Result:



```
East Village
Union Sq
Greenwich Village North
Gramercy
Lower East Side
```

7. NoSQL

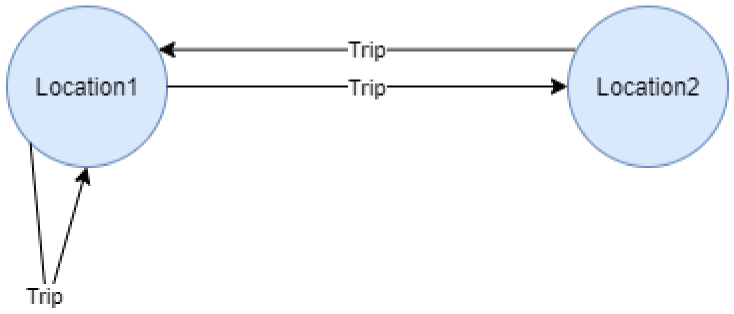
As we started developing our project in MySQL we wondered if MySQL or any other standard SQL Technology will be able to handle this volume of data. So we wanted to try out some NoSQL big data technologies. For this purpose, we used the graph database Neo4j which is a very popular production level database. We also wanted to compare this performance with another document database for which we opted for OrientDB.

Following are the differences between the Technologies used:

MySQL	**Neo4j**	**OrientDB**
Data Stored in the form of rows and columns	Data Stored in the form of vertices and relationships	Data Stored in the form of classes and objects in key value format
Data needs to be structured which can lead to a lot of null values	Data doesn't need to be structured.	Data doesn't need to be structured.
Supports ACID properties	Supports BASE properties	Supports BASE properties

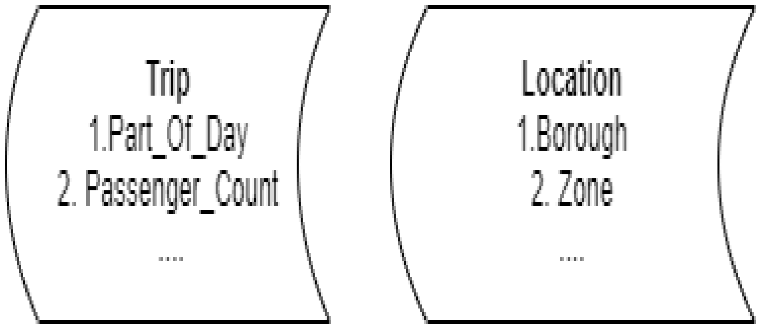
Following are the data models used for the NoSQL databases:

Neo4J:



In Neo4j, the locations are stored as vertices and each trip is stored as a relationship between them. Many relationships start and end at the same vertex as there are many trips starting and ending in the same location.

OrientDB:



In OrientDB, we have created two classes, Trip and Location which are stored in the form of documents. Each class has their own attributes corresponding to the type of entity.

8. NoSQL v/s NoSQL Performance Analysis

Operation	SQL Without Index	SQL with Index on Day of Week, Part of Day, DropOff	NoSQL
Loading Data	15 hours Approx	15 hours Approx	Neo4J ~ 10 min OrientDB ~ 2 min
Running query 3	0.172 sec	0.031 sec	Neo4J ~ 30 sec OrientDB ~ 1 sec
Running Query 3 for a Specific day of week	0.188 sec	0.032 sec	Neo4J ~ 40 sec OrientDB ~ 0.6 sec

9. Future Work

Due to time constraints we were unable to do a few things which we wanted to do. Following are the things which we will take up in the future:

- Add more data for other months to the databases and run the queries on them. We will be able to figure out how the databases stand up to more volume of data.
- Use a different approach in graph database where the trips won't be the relationships between the locations.
- Explore other NoSQL technologies and try to figure out more optimal ones.
- Extend the third Business Question to generate a chain of locations that a driver can visit in order to maximize his utilization while going to his desired locations on time.

10. Conclusion

Summarization and Indexing improved the performance of queries about 6-fold. The queries were able to give the results that are normally expected as per the trend in NYC. And the following conclusions can be drawn from the comparison of SQL with NoSQL technologies:

- MySQL selection performance was better as the data was modelled better for the MySQL database.
- NoSQL databases load the data faster than SQL databases.
- For graph databases, creating multiple relationships of the same type is not optimal.

11. References

Id	Description	Link
1	Taxi and Ride hailing Usage in New York City	https://toddwschneider.com/dashboards/nyc-taxi-ridehailing-uber-lyft-data/
2	Taxi medallion in NYC	https://en.wikipedia.org/wiki/Taxi_medallion#New_York_City
3	NYC TLC Taxi data	https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
4	W3schools tutorial my SQL	https://www.w3schools.com/sql
5	MySQL 8.0 documentation	https://dev.mysql.com/doc
6	Pandas documentation	https://pandas.pydata.org/pandas-docs/stable
7	Neo4j documentation	https://neo4j.com/docs
8	OrientDB documentation	http://orientdb.com/docs