Manhattan Transport Services: Taxis, Subway and FHVs

Mitchell Merrick-Thirlway
New York University
mm12063@nyu.edu

Adidev Vinaik
New York University
av3034@nyu.edu

Anav Prasad
New York University
ap7152@nyu.edu

Abstract

Each weekday, over 2.5 million people use public transportation in NYC[1]. The objective of this project is to discover if there are any trends or insights which can be discovered when comparing different modes of transport commuters choose to use in Manhattan. If insights are discovered, we will interpret how these can be used to better inform City officials so that more sound decisions may be made to help ease congestion for commuters. We will be using publicly available data for NYC's Subway[2], Yellow Taxis[3], and FHVs (Uber/Lyft)[4] for the period of 2015-2021.

1 Introduction

Each weekday, over 2.5 million people use public transportation in NYC. NYC features the world's oldest subway system, one of the most famous taxi services, the NYC Yellow Taxi and enjoys the full benefits of modern private taxis offered by companies like Uber and Lyft. With such a large area to cover and with it being so population dense this city places a great emphasis on the transport available to its residents. To that end, it's important to be able to keep track of their usage and how they can be improved.

This report hopes to shed some light on the situation using all available data over a significant period of time. The goal is to provide insights on high traffic or "busy" zones as well as trends in usage to help officials understand which areas and modes are to be prioritized and to confirm (or debunk) certain hypotheses about usage patterns during different or noteworthy periods of the years being examined.

Commuting can be stressful, time-consuming, and expensive, particularly in large cities with heavy traffic. If City officials can better plan for high congestion for a particular mode of transport at a particular time of the day in a particular part of the city, the experience of the commuters would be greatly improved.

2 Related work / Commuting in cities

In recent years, there has been a growing concern over traffic congestion and its impact on commuters in Manhattan. Various studies and initiatives have been undertaken to address this issue. One study[5] conducted by the New York City Department of Transportation analyzed the impacts of slow moving public transportation (like buses) on traffic congestion in Manhattan. The study found that slow travel speeds affected key corridors beyond the Manhattan core region, and that bus speeds in the outer boroughs fell more noticeably between 2015 and 2017 than they did in the Central Business District in Manhattan. It concluded that NYC's investment in roadway projects were well founded and should be reinforced.

The impact of ride-hailing services on the New York City taxi industry has also been a topic of discussion. The Taxi and Limousine Commission (TLC) conducted a study[6] on this topic, which found that the number of taxi trips in Manhattan had declined significantly since the introduction of ride-hailing services. The study recommended that the City should consider policies to level the playing field between traditional taxis and ride-hailing services, such as implementing minimum wage requirements for ride-hail drivers.

In response to these challenges, the Regional Plan Association proposed[7] a congestion pricing plan for Manhattan, which would charge drivers a fee to enter the most congested parts of the city during peak hours. The plan is aimed at reducing traffic congestion and improving public transportation, and has been implemented in other cities around the world with some success. Finally, a report by the Partnership for New York City highlighted the economic impact of traffic congestion on businesses and residents in Manhattan. The report recommended that the City should prioritize investments in public transportation and implement policies to reduce traffic congestion, such as congestion pricing.

Overall, these studies and initiatives provide valuable insights into the challenges and opportunities for improving transportation in Manhattan. By building upon their findings and recommendations, we seek to contribute to a better understanding of the transportation landscape in Manhattan and inform policies to reduce congestion and improve mobility for commuters.

3 Design Diagram

Figure 1 displays the general structure of our project workflow. We utilize the three main datasets to cover a broad area of public transport in NYC - the Subways, the Yellow Taxis and the For-Hire Vehicles operating in the city. These datasets are collected from various sources and need to be cleaned and reshaped before they can be combined and used for any meaningful inferences. We also need to use some smaller datasets, out of which the NYC Taxi Zone dataset is pivotal as it provides us the common column we can base some of our most impactful results on.

The cleaning is performed using MapReduce and some careful manipulations based on each dataset's columns. After this cleaning is over we have the `Location ID` and `Date` columns to combine our `FHV Trips`, `TLC Trips` and `Transit Trips`.

We use Presto to perform our analyses via queries. Presto was chosen because of its ease of use and adaptability. We ran the SQL files needed to query through Hive by constructing a Python script that would automatically create and run them, saving the output in a text file. Then, we performed our final analyses using Python to plot the charts and graphs needed to make inferences. Since our final data was to be limited to a maximum of 84 columns (12 months for each year) we could easily and quickly plot and draw conclusions.

4 Data Sources and Pre-Processing

4.1 Yellow Taxi TLC Trip Record Data – 80GB

To analyze the Yellow Taxi (YT) trips around Manhattan, the YT TLC Trip Record Data[8] was used. This data set contains over 80GB of trips recorded over the seven year period of 2015-2021. The full list of fields included in this data set can be viewed below and a sample of the data set can be seen in Fig. 2.

The data is quite fine grain in that it has collected minute-by-minute pick and drop-off data. As these files are hosted on the nyc.gov TLC website[8] and split the files split by each month for every year and not available to download in large file, a python script 'yellow_taxi_data_downloader.py' (Fig. 5) was created to automate the downloading process. The downloaded files are in the parquet file format.

- VendorID
- tpep_pickup_datetime
- tpep_dropoff_datetime
- passenger_count
- trip_distance
- RatecodeID
- store_and_fwd_flag
- PULocationID
- DOLocationID
- payment_type
- fare_amount
- extra
- mta_tax
- tip_amount
- tolls_amount
- improvement_surcharge
- total_amount
- congestion_surcharge
- airport_fee

4.2 Data cleaning and pre-processing

After some preliminary viewing of the data locally via python (reading in the parquet files into a Pandas Dataframe and printing head/tail), a MapReduce job (java_yellow_taxi_cleaning/.../ YellowTaxiMapper.java)[9] was created to pre-process and clean the data. This MR job was just a map job, so there was no need for a reducer when pre-processing and used NullWritable

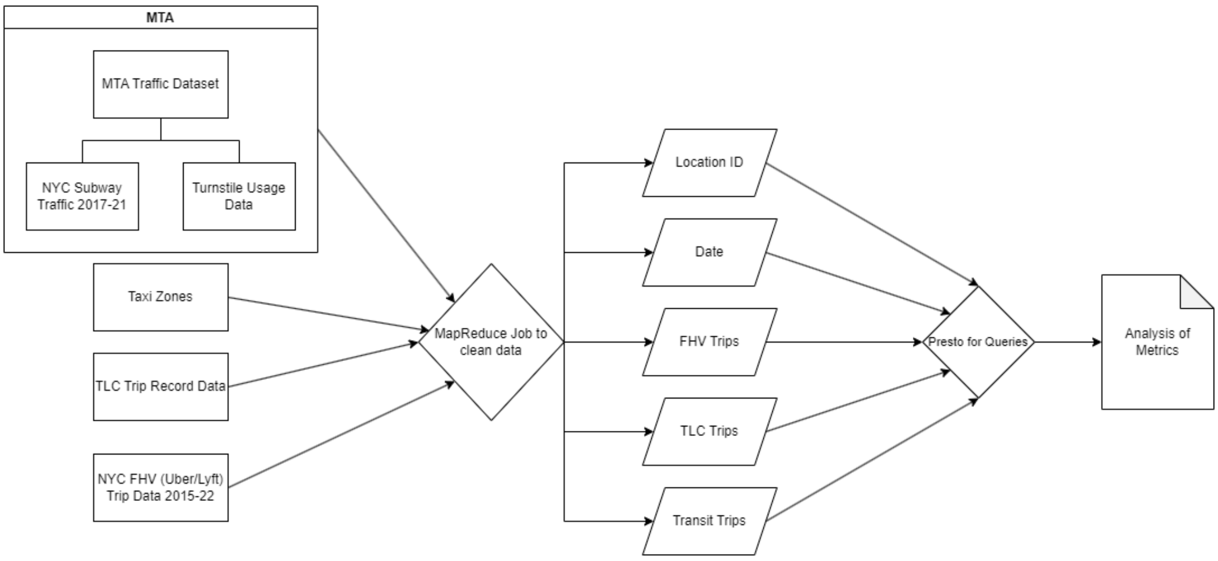
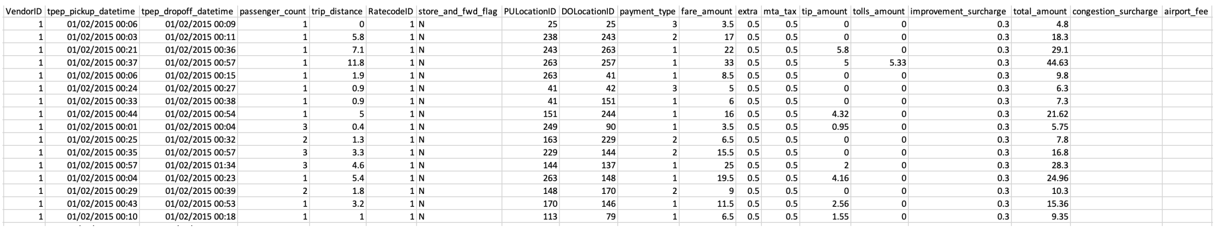


Figure 1: Design Diagram



VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	improvement_surcharge	total_amount	congestion_surcharge	airport_fee
1	01/02/2015 00:06	01/02/2015 00:09	1	0	1	N	25	25	3	3.5	0.5	0.5	0	0	0.3	4.8		
1	01/02/2015 00:03	01/02/2015 00:11	1	5.8	1	N	238	243	2	17	0.5	0.5	0	0	0.3	18.3		
1	01/02/2015 00:21	01/02/2015 00:36	1	7.1	1	N	243	263	1	22	0.5	0.5	5.8	0	0.3	29.1		
1	01/02/2015 00:37	01/02/2015 00:57	1	11.8	1	N	263	257	1	33	0.5	0.5	5	5.33	0.3	44.63		
1	01/02/2015 00:06	01/02/2015 00:15	1	1.9	1	N	263	41	1	8.5	0.5	0.5	0	0	0.3	9.8		
1	01/02/2015 00:24	01/02/2015 00:27	1	0.9	1	N	41	42	3	5	0.5	0.5	0	0	0.3	6.3		
1	01/02/2015 00:33	01/02/2015 00:38	1	0.9	1	N	41	151	1	6	0.5	0.5	0	0	0.3	7.3		
1	01/02/2015 00:44	01/02/2015 00:54	1	5	1	N	151	244	1	16	0.5	0.5	4.32	0	0.3	21.62		
1	01/02/2015 00:01	01/02/2015 00:04	3	0.4	1	N	249	90	1	3.5	0.5	0.5	0.95	0	0.3	5.75		
1	01/02/2015 00:25	01/02/2015 00:32	2	1.3	1	N	163	229	2	6.5	0.5	0.5	0	0	0.3	7.8		
1	01/02/2015 00:35	01/02/2015 00:57	3	3.3	1	N	229	144	2	15.5	0.5	0.5	0	0	0.3	16.8		
1	01/02/2015 00:57	01/02/2015 01:34	3	4.6	1	N	144	137	1	25	0.5	0.5	2	0	0.3	28.3		
1	01/02/2015 00:04	01/02/2015 00:23	1	5.4	1	N	263	148	1	19.5	0.5	0.5	4.16	0	0.3	24.96		
1	01/02/2015 00:29	01/02/2015 00:39	2	1.8	1	N	148	170	2	9	0.5	0.5	0	0	0.3	10.3		
1	01/02/2015 00:43	01/02/2015 00:53	1	3.2	1	N	170	146	1	11.5	0.5	0.5	2.56	0	0.3	15.36		
1	01/02/2015 00:10	01/02/2015 00:18	1	1	1	N	113	79	1	6.5	0.5	0.5	1.55	0	0.3	9.35		

Figure 2: Sample of the (uncleaned) Yellow Taxi data set

as the key. As there were so many parquet files (96 = 12months x 8 years) to process, we used the `MultipleInputs.addInputPath()` method within a loop to pass each parquet file location in the driver code file.

It was decided to keep the following columns and discard the rest as they appeared to contain the most valuable data for finding correlations between the data sets.

- tpep_pickup_datetime – pick-up time
- tpep_dropoff_datetime – drop-off time
- passenger_count
- trip_distance
- PULocationID – Pick up ID
- DOLocationID – Drop off ID
- payment_type – Representative values from 1-6
- total_amount

For the pick up and drop off time, currently stored in the 'DD/MM/YYYY HH:MM' format, it was decided to split this date and time into day, month year, as well as time, for both pick-up and drop-off timestamps. By doing this, it makes it much easier for us to make queries based on the month and year and time of day.

Regarding time of day, three additional columns were introduced: htp_am, htp_pm, trip_period. htp_[am|pm] refer to High Traffic Periods – rush hour(s) in Manhattan during the morning and afternoon. The newly separated time (hour) was used to check whether the trip was taken within one of the rush hour(s) periods (5-10am / 3-7pm) and updated the new columns accordingly with a boolean [1|0]. The trip_period field refers to the [Morning | Afternoon | Evening | Late Evening | Late Night] period of the day so that analytics could be performed on that aspect of the data.

Two more columns were also added: pu_taxi_zone and do_taxi_zone. As the pick up and drop off locations of the trips were recorded using the location IDs of the taxi zones (roughly based on NYC Department of City Planning's Neighborhood Tabulation Areas (NTAs)[10], we needed to join the taxi zones' data set[10] with the YT data set so that we could introduce the human readable English names of each taxi zone. As the taxi zone (TZ) data set was rather small (described in detail later in this report), we were able to make use MapReduce's distributed cache and load the TZ data set into each of the mapper worker nodes (Fig 6) and retrieved the data in the mapper class (Fig 7). The location IDs and readable names were then placed in a hash map so that we could use the location ID in the YT data set as a key reference for the hash map, making it possible to retrieve the readable names in constant time and then adding them to the pu_taxi_zone and do_taxi_zone columns for later use. By loading the TZ data into the worker nodes via the cache, we're reducing the number of reads from HDFS and so improving latency.

The payment_types were kept in case we decide to use this in our future work when comparing our data sets. The values 1-6 related to: 1 = Credit card, 2 = Cash, 3 = No charge, 4 = Dispute, 5 = Unknown, 6 = Voided trip[11] and were converted into their English readable titles for ease of reference. When pre-processing the data set, we ensured that the values for payment_types column used one of these values. For passenger_count we wanted to make sure that it never had a 0 for its value. So if we found a 0, we set it to 1 – we presume at least one person was in the taxi.

The final column fields are listed below:

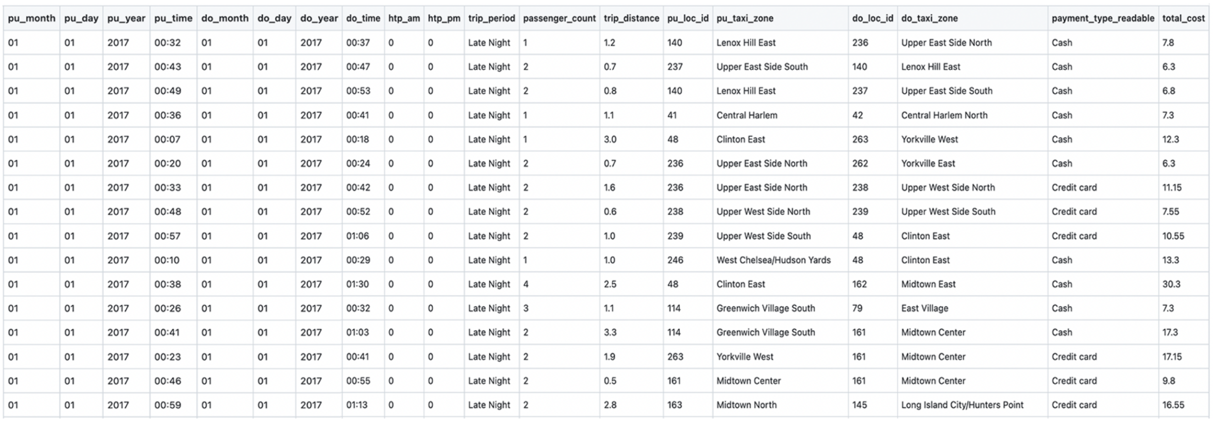
- pu_month
- pu_day
- pu_year
- pu_time
- do_month
- do_day
- do_year
- do_time
- htp_am
- htp_pm
- trip_period
- passenger_count
- trip_distance
- pu_loc_id
- pu_taxi_zone
- do_loc_id
- do_taxi_zone
- payment_type_readable
- total_cost

Having never worked with parquet files before loading them into MR was quite the challenge. Once the parquet files were loaded into MR, however, we set about checking the data set for missing values. An if statement was placed around the emitter which checks for any errors that may have been encountered during the cleaning process. For most of the issues (0 passengers –> 1, we could fix), though, when there were obvious errors like a trip being recorded but the distance was recorded as 0, it decided that that row should be considered an 'error' and not be included in my cleaned data set so updated the value of the error flag variable to ensure the emitter didn't fire. Other serious issues included distances of less than a couple of miles being charged thousands of dollars. These were obviously errors, so the whole row was excluded from the data set.

To obtain the values from the parsed parquet files, we decided to store the column names (and their indices as their values) in an enum class so that we could access them in constant time. Storing the indices and column names saved us from having to remember (or count along the columns) the number value of each columns' index; we could just use a helper function we created to call the column name which would then convert to the required index, which would then return the correct column data.

The output of this mapper class is a 'CSV file(s)' – just MR output files (103 of them) but their contents are formatted as CSVs – for which we created the header schema using some shorter names for the columns we wanted to use. To include this header schema as the first line in each output file, we made use of the setup method and emitted the header before any other data.

Fig. 8 depicts a sample of the processed data set 'CSV' output file(s) showing the new columns and the cleaned data after the MR job had completed.



pu_month	pu_day	pu_year	pu_time	do_month	do_day	do_year	do_time	htp_am	htp_pm	trip_period	passenger_count	trip_distance	pu_loc_id	pu_taxi_zone	do_loc_id	do_taxi_zone	payment_type_readable	total_cost
01	01	2017	00:32	01	01	2017	00:37	0	0	Late Night	1	1.2	140	Lenox Hill East	236	Upper East Side North	Cash	7.8
01	01	2017	00:43	01	01	2017	00:47	0	0	Late Night	2	0.7	237	Upper East Side South	140	Lenox Hill East	Cash	6.3
01	01	2017	00:49	01	01	2017	00:53	0	0	Late Night	2	0.8	140	Lenox Hill East	237	Upper East Side South	Cash	6.8
01	01	2017	00:36	01	01	2017	00:41	0	0	Late Night	1	1.1	41	Central Harlem	42	Central Harlem North	Cash	7.3
01	01	2017	00:07	01	01	2017	00:18	0	0	Late Night	1	3.0	48	Clinton East	263	Yorkville West	Cash	12.3
01	01	2017	00:20	01	01	2017	00:24	0	0	Late Night	2	0.7	236	Upper East Side North	262	Yorkville East	Cash	6.3
01	01	2017	00:33	01	01	2017	00:42	0	0	Late Night	2	1.6	236	Upper East Side North	238	Upper West Side North	Credit card	11.15
01	01	2017	00:48	01	01	2017	00:52	0	0	Late Night	2	0.6	238	Upper West Side North	239	Upper West Side South	Credit card	7.55
01	01	2017	00:57	01	01	2017	01:06	0	0	Late Night	2	1.0	239	Upper West Side South	48	Clinton East	Credit card	10.55
01	01	2017	00:10	01	01	2017	00:29	0	0	Late Night	1	1.0	246	West Chelsea/Hudson Yards	48	Clinton East	Cash	13.3
01	01	2017	00:38	01	01	2017	01:30	0	0	Late Night	4	2.5	48	Clinton East	162	Midtown East	Cash	30.3
01	01	2017	00:26	01	01	2017	00:32	0	0	Late Night	3	1.1	114	Greenwich Village South	79	East Village	Cash	7.3
01	01	2017	00:41	01	01	2017	01:03	0	0	Late Night	2	3.3	114	Greenwich Village South	161	Midtown Center	Cash	17.3
01	01	2017	00:23	01	01	2017	00:41	0	0	Late Night	2	1.9	263	Yorkville West	161	Midtown Center	Credit card	17.15
01	01	2017	00:46	01	01	2017	00:55	0	0	Late Night	2	0.5	161	Midtown Center	161	Midtown Center	Credit card	9.8
01	01	2017	00:59	01	01	2017	01:13	0	0	Late Night	2	2.8	163	Midtown North	145	Long Island City/Hunters Point	Credit card	16.55

Figure 3: Pre-processed Yellow Taxi data set

4.3 Analysis of the Yellow Taxi Data Set

Once the data had been cleaned and the values formatted as we required, MR jobs were created to run some basic statistic analysis on the data set. Due to the size of the data, however, MR struggled to process all 571 million rows of data; took too long to process and invariably experienced Java heap space errors. As a result, we loaded the CSV output files into Hive so that we could run Presto SQL commands on the data to perform the analysis. As we wanted to run many SQL commands, we created a Python script (sql_query_file_creator.py[12]) to automate the creation of SQL files. Once we had created these SQL files, we used the Presto CLI tool to parse the SQL file and out the results to a txt file for us to use in analysis: `presto -catalog hive -schema default -file sql_files/{FILE_NAME}.sql sql_output/{FILE_NAME}_output.txt`. Examples of the SQL files we created can be found in `/sql_files/`[13].

Basic statistics (over 2015-2017)

- Total Trips: 570,932,836
- Avg Passenger Count: 1.62
- Avg Cost: 14.83
- Trip Distance Min: 0.01
- Trip Distance Max: 17.4
- Trip Distance Avg: 1.82

To see if/how the usage of YTs and the costs had changed over the years, we created Presto SQL commands to retrieve the average cost and distance for each year (cost_avg_per_year.sql, dist_avg_per_year.sql[13]). The results are depicted in Fig. 4. We can see that, the average distance hasn't changed that much over the years moving between 1.78 miles and 1.93. The cost of using a YT, however, has gone up with a dramatic jump of around $2.50 a ride in 2018.

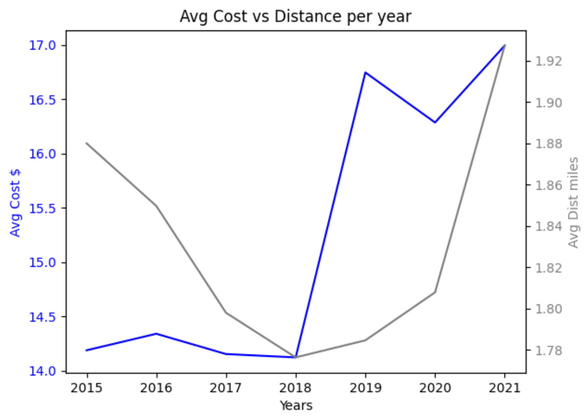
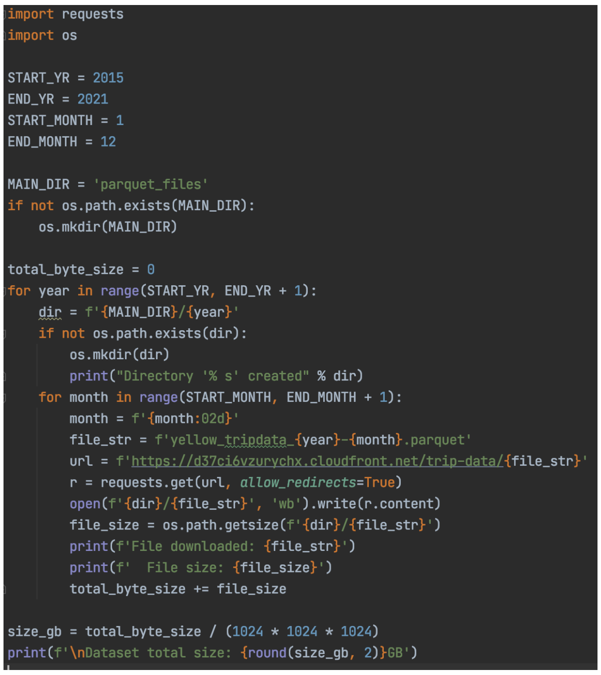


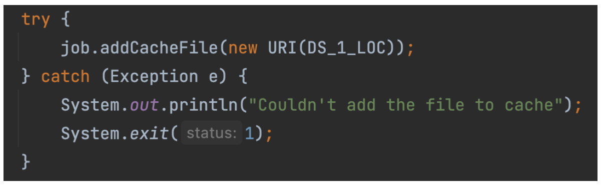
Figure 4: Avg Costs and Distance over the years

Once we had these basic statistics, we wanted to find the top 10 pickup locations over 2015-2021 to see where YTs are most popular. To do this, we created another job in YellowTaxiStats called YellowTaxiStatsCountMapper[14]. The role of this job was to search through each trip and count the number of times each location was seen (basically WordCount, using the location name as the key). Once this was completed, we created another job directly after it which used the output file from the YellowTaxiStatsCountMapper job. This new job, YellowTaxiStatsTop10[15], used the 'Top10' design pattern to process the input file and display the results.



```python
import requests
import os

START_YR = 2015
END_YR = 2021
START_MONTH = 1
END_MONTH = 12

MAIN_DIR = 'parquet_files'
if not os.path.exists(MAIN_DIR):
    os.mkdir(MAIN_DIR)

total_byte_size = 0
for year in range(START_YR, END_YR + 1):
    dir = f'{MAIN_DIR}/{year}'
    if not os.path.exists(dir):
        os.mkdir(dir)
        print("Directory '% s' created" % dir)
    for month in range(START_MONTH, END_MONTH + 1):
        month = f'{month:02d}'
        file_str = f'yellow_tripdata_{year}-{month}.parquet'
        url = f'https://d37ci6vzurychx.cloudfront.net/trip-data/{file_str}'
        r = requests.get(url, allow_redirects=True)
        open(f'{dir}/{file_str}', 'wb').write(r.content)
        file_size = os.path.getsize(f'{dir}/{file_str}')
        print(f'File downloaded: {file_str}')
        print(f'   File size: {file_size}')
        total_byte_size += file_size

size_gb = total_byte_size / (1024 * 1024 * 1024)
print(f'\nDataset total size: {round(size_gb, 2)}GB')
```

Figure 5: Python script used to download the parquet files



```java
try {
    job.addCacheFile(new URI(DS_1_LOC));
} catch (Exception e) {
    System.out.println("Couldn't add the file to cache");
    System.exit( status: 1);
}
```

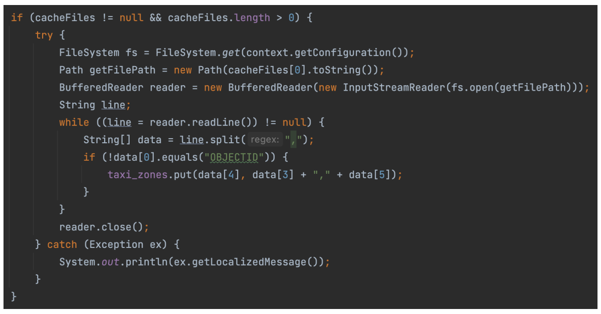
Figure 6: Adding the Taxi Zones data set to the distributed cache

We now became curious as to how YTs have been used over the years so we wanted to find the top 10 pickup locations for each year to see if there was any obvious migration of people who use YTs around the city. Fig 15 through Fig 21 (Appendix A) show the top 10 pickup locations for each year. As you can see, the Upper East Side has consistently been the most popular area for commuters to use YTs and has grown even more popular over the years.

To see how these YTs are being used, we wanted to investigate the most popular trips over the years. To do this we created MR jobs with the pick up and drop off locations concatenated to create the key. We then counted the number of occurrences to produce the results depicted in Fig. 9 for 2021.

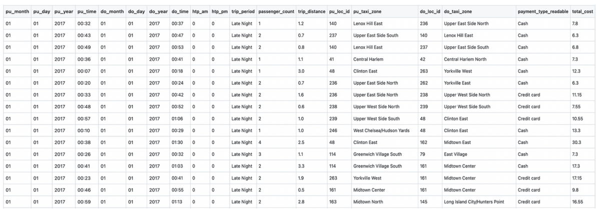
Again, we can see the Upper East Side is a hotbed of YT activity, with most trips occurring within the same area of Manhattan!

Out of curiosity, we also looked into how the methods of payment have changed over the years.



```java
if (cacheFiles != null && cacheFiles.length > 0) {
    try {
        FileSystem fs = FileSystem.get(context.getConfiguration());
        Path getFilePath = new Path(cacheFiles[0].toString());
        BufferedReader reader = new BufferedReader(new InputStreamReader(fs.open(getFilePath)));
        String line;
        while ((line = reader.readLine()) != null) {
            String[] data = line.split( regex: ",");
            if (!data[0].equals("OBJECTID")) {
                taxi_zones.put(data[4], data[3] + "," + data[5]);
            }
        }
        reader.close();
    } catch (Exception ex) {
        System.out.println(ex.getLocalizedMessage());
    }
}
```

Figure 7: Loading the Taxi Zone data into the mapper class



pu_month	pu_day	pu_year	pu_time	do_month	do_day	do_year	do_time	htp_am	htp_pm	trip_period	passenger_count	trip_distance	pu_loc_id	pu_taxi_zone	do_loc_id	do_taxi_zone	payment_type_readable	total_cost
01	01	2017	00:32	01	01	2017	00:37	0	0	Late Night	1	1.2	140	Lenox Hill East	236	Upper East Side North	Cash	7.8
01	01	2017	00:43	01	01	2017	00:47	0	0	Late Night	2	0.7	237	Upper East Side South	140	Lenox Hill East	Cash	6.3
01	01	2017	00:49	01	01	2017	00:53	0	0	Late Night	2	0.8	140	Lenox Hill East	237	Upper East Side South	Cash	6.8
01	01	2017	00:36	01	01	2017	00:41	0	0	Late Night	1	1.1	41	Central Harlem	42	Central Harlem North	Cash	7.3
01	01	2017	00:07	01	01	2017	00:18	0	0	Late Night	1	3.0	48	Clinton East	263	Yorkville West	Cash	12.3
01	01	2017	00:20	01	01	2017	00:24	0	0	Late Night	2	0.7	236	Upper East Side North	262	Yorkville East	Cash	6.3
01	01	2017	00:33	01	01	2017	00:42	0	0	Late Night	2	1.6	236	Upper East Side North	238	Upper West Side North	Credit card	11.15
01	01	2017	00:48	01	01	2017	00:52	0	0	Late Night	2	0.6	238	Upper West Side North	239	Upper West Side South	Credit card	7.55
01	01	2017	00:57	01	01	2017	01:06	0	0	Late Night	2	1.0	239	Upper West Side South	48	Clinton East	Credit card	10.55
01	01	2017	00:10	01	01	2017	00:29	0	0	Late Night	1	1.0	246	West Chelsea/Hudson Yards	48	Clinton East	Cash	13.3
01	01	2017	00:38	01	01	2017	01:30	0	0	Late Night	4	2.5	48	Clinton East	162	Midtown East	Cash	30.3
01	01	2017	00:26	01	01	2017	00:32	0	0	Late Night	3	1.1	114	Greenwich Village South	79	East Village	Cash	7.3
01	01	2017	00:41	01	01	2017	01:03	0	0	Late Night	2	3.3	114	Greenwich Village South	161	Midtown Center	Cash	17.3
01	01	2017	00:23	01	01	2017	00:41	0	0	Late Night	2	1.9	263	Yorkville West	161	Midtown Center	Credit card	17.15
01	01	2017	00:46	01	01	2017	00:55	0	0	Late Night	2	0.5	161	Midtown Center	161	Midtown Center	Credit card	9.8
01	01	2017	00:59	01	01	2017	01:13	0	0	Late Night	2	2.8	163	Midtown North	145	Long Island City/Hunters Point	Credit card	16.55

Figure 8: Cleaned output Yellow Taxi data after cleaning

This, however, produced no interesting results as credit card, unsurprisingly, has been the most popular payment type for every year.

We wanted to find out if more trips were being taken during rush hour(s) or outside of them. Fig. 10 shows there were a lot more trips taken during rush hour.

We then wanted find out if more trips were being taken in the AM or PM rush hour(s). Fig. 11 shows that the PM rush hour(s) is definitely more popular.

4.4 NYC FHV (Uber/Lyft) Trip Data - 5GB

This data was obtained from the New York City Taxi & Limousine Commission's (TLC) website. The For-Hire Vehicle or FHV data records the trips for said types of vehicles and these records were obtained from Trip Record submissions made by bases. This data can be used to create a picture regarding the trips made throughout the city.

This data is stored in the Apache Parquet format,

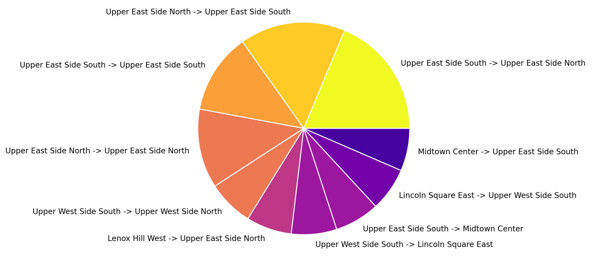


Figure 9: Top 10 trips in 2021

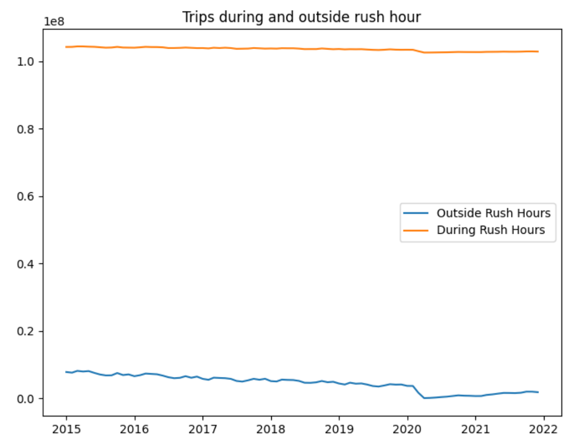


Figure 10: Journeys during and outside of rush hour(s)

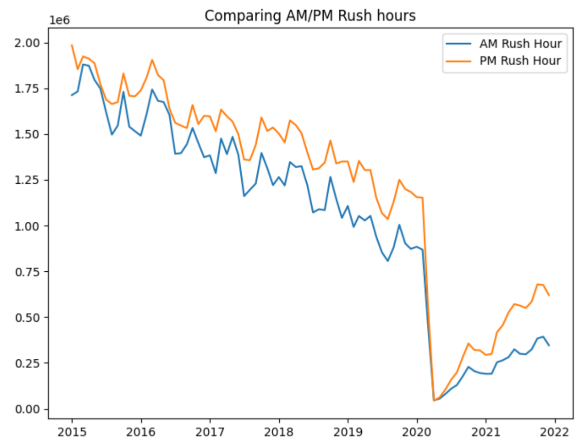
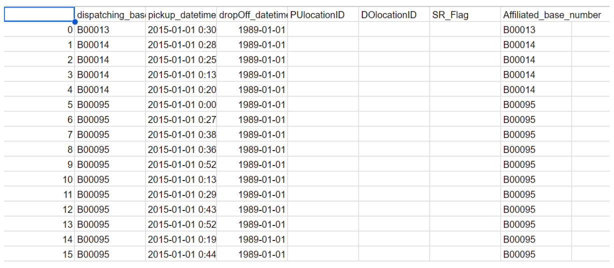


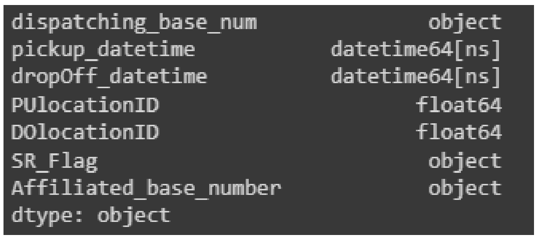
Figure 11: Showing the difference in AM/PM rush hour(s)

which is an open source, column-oriented data file format designed for efficient data storage and retrieval [16]. The compressed nature of this file type ensures that our 706 million records across the 7 years of our study (2015-2021) only take 5 GB of space on disk. We can see the general structure of the data contained in Figure 12. The fields and the data types of each can be seen in Figure 13.



	dispatching_bas	pickup_datetime	dropOff_datetim	PUlocationID	DOlocationID	SR_Flag	Affiliated_base_number
0	B00013	2015-01-01 0:30	1989-01-01				B00013
1	B00014	2015-01-01 0:28	1989-01-01				B00014
2	B00014	2015-01-01 0:25	1989-01-01				B00014
3	B00014	2015-01-01 0:13	1989-01-01				B00014
4	B00014	2015-01-01 0:20	1989-01-01				B00014
5	B00095	2015-01-01 0:00	1989-01-01				B00095
6	B00095	2015-01-01 0:27	1989-01-01				B00095
7	B00095	2015-01-01 0:38	1989-01-01				B00095
8	B00095	2015-01-01 0:36	1989-01-01				B00095
9	B00095	2015-01-01 0:52	1989-01-01				B00095
10	B00095	2015-01-01 0:13	1989-01-01				B00095
11	B00095	2015-01-01 0:29	1989-01-01				B00095
12	B00095	2015-01-01 0:43	1989-01-01				B00095
13	B00095	2015-01-01 0:52	1989-01-01				B00095
14	B00095	2015-01-01 0:19	1989-01-01				B00095
15	B00095	2015-01-01 0:44	1989-01-01				B00095

Figure 12: FHV Columns



```
dispatching_base_num                object
pickup_datetime             datetime64[ns]
dropOff_datetime            datetime64[ns]
PUlocationID                       float64
DOlocationID                       float64
SR_Flag                             object
Affiliated_base_number              object
dtype: object
```

Figure 13: FHV Datatypes

The files are ordered and partitioned by month and year, thus making it easier to parse sections for analysis of a certain time period. We downloaded it from Kaggle as a zip file containing the parquet files. This zip contained data for the year 2022 as well but owing to limitations posed by the overlapping periods of our three major datasets we decided to narrow our observations till the year 2021. Thus we used 84 of the 96 files available to us. This, combined with the NYC Taxi Zones data discussed in the next section, would be the basis of our analysis regarding FHVs.

4.4.1 Column Information

This section details the columns present in our NYC FHV Dataset and the information represented through the data in said columns before cleaning. The columns are as follows:

- **Column:** `dispatching_base_num`
  Description: The TLC Base License Number of the base that dispatched the trip (e.g. Lyft has base license numbers B02510 and B02844).

- **Column:** `pickup_datetime`
  Description: The date and time of the trip pick-up. Represented in the column in the 'YYYY-MM-DD HH:MM:SS' format.

- **Column:** `dropOff_datetime`
  Description: The date and time of the trip dropoff. Represented in the column in the 'YYYY-MM-DD HH:MM:SS' format.

- **Column:** `PUlocationID`
  Description: TLC Taxi Zone in which the trip began. The entry is a number between 1 and 265.

- **Column:** `DOlocationID`
  Description: TLC Taxi Zone in which the trip ended. The entry is a number between 1 and 265.

- **Column:** `SR_Flag`
  Description: Indicates if the trip was a part of a shared ride chain offered by a High Volume FHV company (e.g. Uber Pool, Lyft Line). For shared trips, the value is 1. For non-shared rides, this field is `null`.

- **Column:** `Affiliated_base_number`
  Description: The TLC Base License Number of the base that the vehicle is associated with. This can be the same or different from the dispatch base number.

4.4.2 Data Cleaning and Pre-Processing

We required some amount of cleaning and pre-processing to make any meaningful inferences from this dataset. With 706,893,667 rows, each representing a single trip, we had a lot of data to work with and decided to compress it by month. Our end result would be 84 rows of data that we could then use alongside the other datasets and run quick queries on. The following steps were taken to achieve this:

- **Malformed Row Removal**: When observing the dataset rows (checked in Python using pyarrow before we moved to HDFS) we noticed that the `dropOff_datetime` row had certain entries that corresponded to impossible dates (like 1989-01-01) which were clearly dummy entries. Furthermore, the `PUlocationID` and `DOlocationID` columns had `NaN` entries which were not permissible. The MapReduce job used to clean the dataset would simply check if this value existed by matching the column index using the `getValuetoString` function. If we did not get a valid value we would not write the row to the output.

- **Transforming and Dropping columns**: The `SR_Flag` column also used `null` as a possible value, which isn't very useful for reading and analysis in HDFS. The best way to deal with this would be transform it to another type of value (0 in place of `null` for example) but considering the next steps we decided to drop all unnecessary columns. In this step, which was performed by not including these columns in the output for the Map Reduce cleaning job, we drop the `dispatching_base_num`, `DOlocationID`, `SR_Flag` and `Affiliated_base_number` columns as they pose no relevance to our analysis.

- **Adding columns**: This step is more complicated than it initially appears. We use the cache to store the relatively small NYC Taxi Zone dataset (266 rows) and match that to the `PUlocationID`, getting the `Borough` and `Zone` columns from it. This will be needed for future steps. Along with that we split the `pickup_datetime` column into `day`, `month`, `year` and `time` columns. The time column is then used to check whether the trip counts as happening during either rush hour period.

  We add two columns with boolean values to indicate whether it falls into the 5 to 10 AM window (`am_rush`) or 3 to 7 PM window (`pm_rush`). The `pickup_datetime` column can still be used as the key for this data in this step.

- **Aggregation**: Now, to perform our analysis, we need to aggregate this data, since it still constitutes a large number of rows and is extremely varied based on timestamps. To do this we decide to count the number of trips per month. We could use the year and month to check each new row. This count could be performed by simply noting the values in the `year` and `month` columns, passing that as key with the number 1 per trip.

  However, this was not the only kind of aggregation we needed so we performed the same task but also included location as part of the key. The idea was that we would want to see the number of trips for each location to make inferences on that as well. This could be summed for a time period using the aforementioned columns by passing the data through a query and we can get a combined result for the year and month key.

Furthermore, we also wanted to check our results for a particular time period - during COVID and for each season. This counting was performed using the queries used for the Yellow Taxi dataset.

4.4.3 Analysis of the FHV Data Set

Our analysis was performed on the cleaned data. The procedure was similar to the one adopted for the Yellow taxi dataset. The only other quantity we decided to examine was the average number of trips per year to try to check if the data showed any trends that would be useful in our next section. The averages are given below:

- 2015 - 5282377.667
- 2016 - 11009506.917
- 2017 - 16025796.5
- 2018 - 21739562.75
- 2019 - 3605106.333
- 2020 - 1245455.417

We can clearly see the impact the COVID-19 pandemic had on the FHVs by the average trips. This is further examined in the next sections.

4.5 MTA Turnstile Traffic Data - 5.5 GB

MTA Turnstile Traffic Data[2] was collected from NYS Open Data Program and merged to get data in the time period that was desired (2015 – 2021)(google drive link).

For pre-processing of the MTA Turnstile traffic data, a few more datasets were required. They are listed as follows:

- NYC Borough Boundaries[17]
  Perimeter polygonal boundaries of NYC boroughs in GPS coordinates
- NYC Taxi Zones[18]
  List of taxi zone location id along with its perimeter polygonal shape in GPS coordinates, borough, etc.
- NYC Subway Stations[19]
  List of NYC Subway Stations along with their location in GPS coordinates, station names, etc.

4.5.1 Pre-Processing of Supplementary Datasets

NYC Borough Boundaries

This dataset[17] has 5 rows (for the five boroughs) and 5 columns (`the_geom` (polygon), `BoroCode` (borough code), `BoroName` (borough name), `Shape_Leng` (shape length), `Shape_Area` (shape area)). Only one thing from this dataset was required that is the Manhattan polygonal perimeter in GPS coordinates so that the list of subways stations could be filtered down to just Manhattan (discussed further on). As such, the rest of the data in this dataset was discarded.

NYC Taxi Zones

NYC Taxi Zones data[18] was obtained so as to have a common key (taxi zone location id) across all the major datasets (Yellow taxi, MTA traffic and, FHV data) so that they could eventually be analyzed. It had 264 rows and 7 columns (`OBJECTID`, `Shape_Leng` (shape length), `the_geom` (polygonal perimeter in GPS coordinates), `Shape_Area` (shape area), `zone` (taxi zone name), `LocationID` (taxi zone location id), `borough` (borough name).
Using the borough column, this data was filtered down to Manhattan taxi zones. Note that the `the_geom` column gave the polygonal perimeter of the taxi zones by GPS coordinates.

NYC Subway Stations

This dataset[19] has 6 columns (`URL` (url), `OBJECTID` (object id), `NAME` (station name), `the_geom` (polygonal perimter), `LINE` (subway lines), `NOTES`) and 473 rows. It contained a list of NYC subway stations across all boroughs. It is to be noted that the `OBJECTID` here is different from the remote unit ID (`UNIT` column) in MTA Turnstile Traffic Data. The following steps were taken for pre-processing of this data:

- First of all, a polygon-point algorithm[20] and Manhattan borough polygonal perimeter boundary data was used to filter the list of subway stations across all boroughs down to just Manhattan. This algorithm functions by analyzing if a given point exists inside or outside a given polygon. As such, the perimeter multi polygon of Manhattan in GPS coordinates and, the GPS coordinates of the subway stations allowed the algorithm to give the requisite results.

- Secondly, the filtered list of stations was manually annotated with its remote unit ID (of MTA Turnstile Traffic Data) using matching station names and subway lines passing through the respective stations. Manual annotation was required due to a lack of a dataset that connected the object ids mentioned in NYC Subway Stations data with remote unit IDs in MTA Turnstile Traffic data. Station names could not be used to join these two datasets because of redundancy among the names.

- Finally, the polygon-point algorithm[20] was used again to annotate the subway stations with their respective taxi zone location IDs using NYC Taxi Zones data[18].

4.5.2 Data Structure of MTA Turnstile Traffic Data

The MTA Turnstile traffic data, before cleaning, had 11 columns and $68,286,309$ rows. The columns are as follows:

- **Column:** `C/A`
  Description: Control Area name/Booth name. This is the internal identification of a booth at a given station.

- **Column:** `Unit`
  Description: Remote unit ID of station.

- **Column:** `SCP`
  Description: Subunit/Channel/position represents a specific address for a given device.

- **Column:** `Station`
  Description: Name assigned to the subway station by operations planning.

- **Column:** `Line Name`
  Description: Train lines stopping at this location.

- **Column:** `Division`
  Description: Represents the Line originally the station belonged to BMT, IRT, or IND.

- **Column:** `Date`
  Description: Represents the date of the audit data.

- **Column:** `Time`
  Description: Represents the time of the reported data (HH:MM:SS).

- **Column:** `Description`
  Description: Represents the "REGULAR" (regular scheduled; every four hours) or "RECOVR AUD" (missed /recovery) audit event.

- **Column:** `Entries`
  Description: The cumulative ENTRY register value for a device. This register was initialized during system setup.

- **Column:** `Exits`
  Description: The cumulative EXITS register value for a device. This register was initialized during system setup.

Note that entries and exits values are respective cumulative values since a particular device's setup or reset. The first three columns, `C/A`, `Unit` and `SCP` are needed to identify a particular turnstile in a booth at a MTA Station. To get the number of entries/exits in a particular time period, one needs to get the difference between record of entries/exits at the ending timestamp of the period with the record at the starting timestamp of the period.

4.5.3 Data Pre-Processing

For data pre-processing, the following steps were taken:

- **Malformed Row Removal:**
  Malformed rows such as empty rows or rows with wrong data in time, date, entries, exits columns were removed. In the Map Reduce job for this, only mapper was needed and malformed rows were just omitted from being written.

- **Filtering rows to discard data outside Manhattan:**
  The processed NYC Subway Stations data i.e., list of subway stations in Manhattan annotated with remote unit station IDs and taxi zones was used to filter MTA Turnstile traffic data to discard data of subway stations lying outside of Manhattan.
  For this, the processed NYC Subway Stations data, comprising of 147 rows and 4 columns, was read prior to the execution of the mapper and stored in a configuration property to be read back using the context object later

on in the mapper. Again, only mapper was needed in this case and the rows representing subway stations outside Manhattan were just not written. The GPS coordinate of the stations (present in the processed NYC Subway Stations data), was appended as a column to the output.

- **Computation of entries/exits per day per turnstile device (in a booth / control area at a station):**
To get the count of entries/exits for a turnstile device in a booth at a station for a given day, the highest value of entries/exits recorded for a day was subtracted with the lowest (since the values of entries/exits are cumulative). This was done by keeping booth id appended with remote station ID, SCP (unique identifier of a turnstile device in a booth at a station) and date as the output key of the mapper. Then, in the reducer, the lowest value was subtracted from the highest value, for both entries and exits, to get the required values.

- **Computation of entries/exits per day per station:**
Since there can be multiple control areas/booths in a station and multiple turnstile devices per booth, to get the total number of entries/exits in a station, the counts of entries/exits were summed. This was done by keeping the station remote ID, appended with the date, as the output key of the mapper. The counts of entries/exits were then summed up in the reducer for each station for a given day.

- **Addition of taxi zone label column to the data:**
The taxi zones data, which contains the polygonal perimeter data of the zones in GPS coordinates, was used to append the relevant taxi zone location ID as a column in this step. To do this, similar to the third last step, the taxi zones data was read and stored in the configuration and retrieved as needed in the mapper job using the context object. The polygon-point algorithm[20] discussed before was used to find the correct taxi zone location id for each row of the data.

- **Computation of entries/exits per day per taxi zone:**
In the final step, the count of entries/exits for each taxi zone for each day was computed. This was done by keeping the taxi zone location id appended with the date as the output key of the mapper task and the counts of entries/exits were summed up in the reducer task and written.

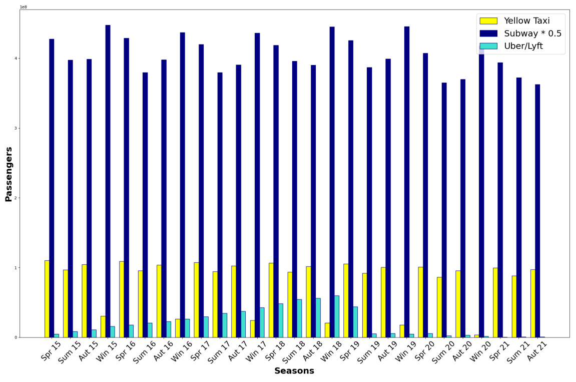


Figure 14: Seasonal Results for YTs

5 Results

5.1 Seasonal Data

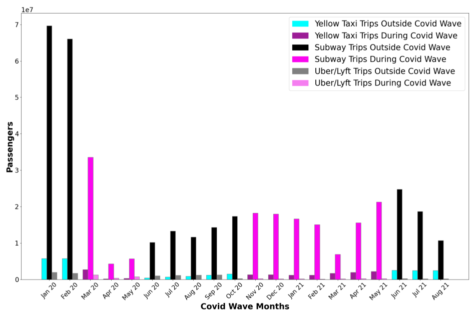
When comparing all three modes of transport we decided to view it per season over the years. Fig. 14 depicts the YT, Subway and FHVs per season. It is very obvious to see that the subway is the most popular mode of transport in Manhattan and is dwarfs the other modes. There are some very obvious trends in the data too, with the subway being used even more in the winter months. The YT data also shows trends with it becoming less popular in the summer months and dropping dramatically in the winter months.

Our initial thoughts on this low trade period was that people didn't want to wait on the street for YTs in the cold weather. On further reflection, however, we believe the more reasonable explanation is that through our investigative work on the YT data, we know the UES is its biggest customer base. As the residents of the UES, who can afford to take YTs, are also those who can afford to escape the winter months in NYC ('snowbirds') and fly to warmer climes of the south (Florida, in particular), we believe the dramatic drop in customers in the winter months is due to fewer customers actually being on the UES.

The FHVs results also so some interestings trends. First, we can see great growth in the use of FHVs from Spring 2015 to Winter 2018 and then a dramatic drop off. At first, we thought this was due to the impact of Covid-19, but then realised that the dats don't actually align with the CV19 pandemic. After some, research we discovered that the minimum wage law for FHVs was introduced. "The minimum wage rule goes into effect in 20 days [Dec 2018] and requires companies to pay drivers at least $27.86 per hour, or $17.22 per hour after expenses."[21] We then see in May 2019, "Uber and Lyft Have Officially Stopped Hiring Drivers in NYC"[22]. This ruling obviously had a huge impact on the busniess model for FHV. It appears as thought the companies stopped supporting their endeavours to expand into Manhattan and as a result, the number of FHV trips died off.

5.2 Covid Data

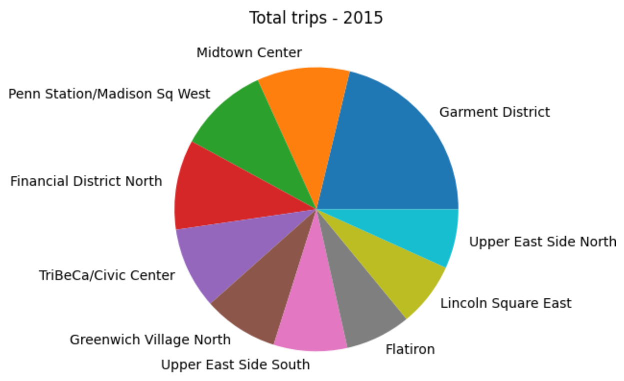
Since the time period of our data covered the COVID-19 pandemic, we also analyzed the effect of the same. The following graph covers two major covid waves[23] of 2020 and 2021.

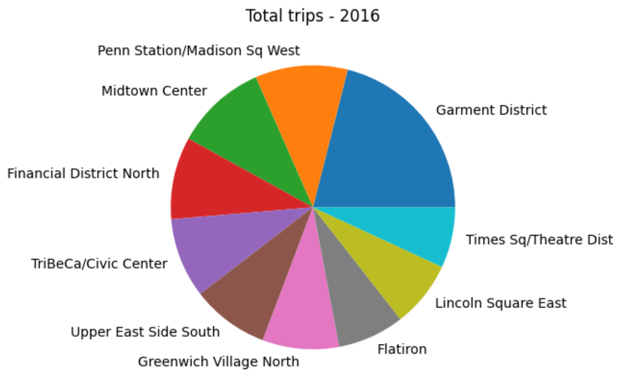


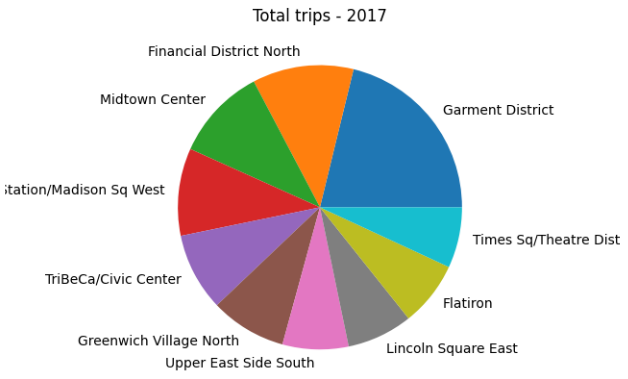
As seen above, the pink regions representing the covid waves, a drastic reduction in usage of the three forms of public transportation can be observed especially across February, March and April of 2020.

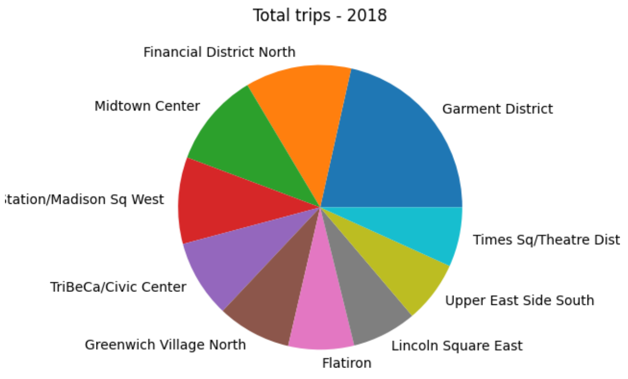
5.3 Trips Data

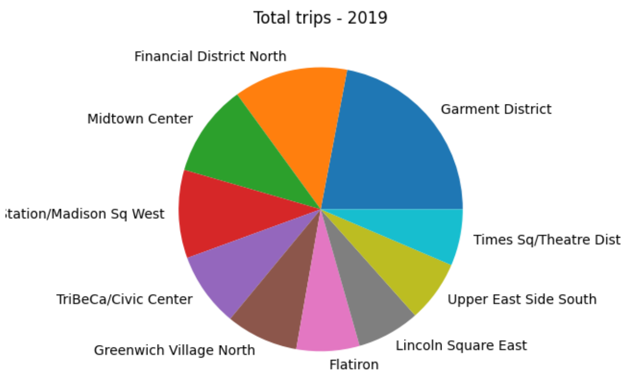
Our goal for this analysis was to examine the number of trips from the perspective of each starting location. We wanted to check which locations would end up being the busiest and what the trend would look like over a period of more than half a decade. The following pie charts show us the top 10 locations with the highest trip count for each year, starting from 2015 to 2022. These trips are a combined value obtained from adding all the types of trips from a location, denoted in the table by a common location ID.

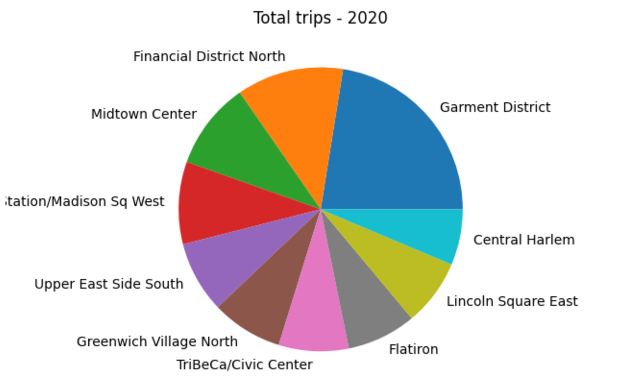


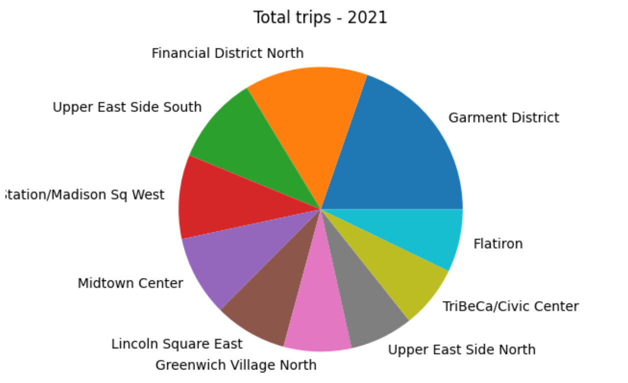












It is pretty evident just by glancing at the data distribution that Garment District is by the busiest location, for all years. It notably has almost double the number of trips as the second highest location. Furthermore, the same 10 locations dominate the top 10 busiest locations throughout. We believe that the cause for this consistency lies in the counts of the underlying data. Since the subway trip count data far exceeds that of the other two types we see the areas with busy subways coming out on top.

The Garment District is located between 34th and 41st Streets, west of 6th Avenue. This range covers high density hotspots and lies extremely close to Times Square and Port Authority, two prominent areas. Our understanding is that this is the cause for it being first throughout our time slice.

6 Goodness

To ensure our data sets and queries were in-sync we based our queries on the timestamps of the trips. By splitting out the timestamps in to years, months, days, we were able to query each data set by year and month, ensuring we were all querying the same periods for comparison. As FHVs and YT data sets both used the Taxi Zone location IDs for their pick up and drop off locations, by joining the Taxi Zone data set into botht he FHV and YTs data sets, we were able to make sure the team we all referring to the same PU and DO locations.

Though more difficult, the TZ data set was used to create a zone polygon via its longitude and latitude data to then plot the Manhattan subway stations within those zones, therby ensuring all data sets were referring to the same taxi zones when comparing trips.

For the Covid-19 wave data, we referred to the NYC Health offical data ranges for the first and second waves of CV19 in NYC.

7 Conclusion

With the rush hour periods being the most popular time of the day for commuting, the subway being by far the most commonly used mode of transport and the Garment District (GD) taxi zone being the more popular area for trips to start, we recommend that City officials invest more heavily in the Garment District. We recommend alternative modes of transport, which are cheaper than FHVs and YTs, to help take away some of the burden on the GD taxi zone, especially for the subway.

We feel this project was a very worthwhile experience. Not only for the learnings above but also to have hands-on experience of using big data analytic tools like MapReduce and Presto.

References

[1] Subway bus ridership 2021. "https://new.mta.info/agency/new-york-city-transit/subway-bus-ridership-2021.

[2] NYC mta turnstile traffic data. https://data.ny.gov/browse?q=turnstile%20usage%20data&sortBy=relevance.

[3] NYC yellow taxi trip records. https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page.

[4] NYC fhv (uber/lyft) trip data. https://www.kaggle.com/datasets/jeffsinsel/nyc-fhv-uberlyft-trip-data-simple-2015-2022.

[5] NYC DOT mobility report 2018. https://www.nyc.gov/html/dot/downloads/pdf/mobility-report-2018-print.pdf.

[6] NYC TLC annual report 2018. https://www.nyc.gov/assets/tlc/downloads/pdf/annual_report_2018.pdf.

[7] Regional Plan Association: Save our Subways. https://s3.us-east-1.amazonaws.com/rpa-org/pdfs/RPA-Save-Our-Subways.pdf.

[8] Tlc trip record data. https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page.

[9] Mapper for yellow taxi dataset. https://github.com/mm12063/rbda-project-spring-23/blob/main/java_yellow_taxi_cleaning/src/main/java/taxi/YellowTaxiMapper.java.

[10] Nyc taxi zones. https://data.cityofnewyork.us/Transportation/NYC-Taxi-Zones/d3c5-ddgc.

[11] Nyc trip records dictionary. https://www.nyc.gov/assets/tlc/downloads/pdf/data_dictionary_trip_records_yellow.pdf.

[12] Sql query file creator. https://github.com/mm12063/rbda-project-spring-23/blob/main/sql_query_file_creator.py.

[13] Dir of sql files. https://github.com/mm12063/rbda-project-spring-23/tree/main/sql_files.

[14] Counter mapper for yellow taxi data. https://github.com/mm12063/rbda-project-spring-23/blob/main/java_yellow_taxi_stats/src/main/java/taxi/YellowTaxiStatsCountMapper.java.

[15] Mapper to determine top 10 areas". https://github.com/mm12063/rbda-project-spring-23/blob/main/java_yellow_taxi_stats/src/main/java/taxi/YellowTaxiStatsTop10Mapper.java.

[16] Apache parquet. https://parquet.apache.org/.

[17] NYC borough boundaries. https://data.cityofnewyork.us/d/tqmj-j8zm?category=City-Government&view_name=Borough-Boundaries.

[18] NYC taxi zones. https://data.cityofnewyork.us/d/d3c5-ddgc?category=Transportation&view_name=NYC-Taxi-Zones.

[19] NYC subway stations. https://data.cityofnewyork.us/d/arq3-7z49?category=Transportation&view_name=Subway-Stations.

[20] Algorithm to determine if a point lies inside or outside a polygon. https://www.geeksforgeeks.org/how-to-check-if-a-given-point-lies-inside-a-polygon/.

[21] Minimum wage law. https://www.thedrive.com/news/25324/new-york-city-votes-to-establish-minimum-wage-for-uber-

[22] Minimum wage law update. https://www.thedrive.com/news/27756/uber-and-lyft-have-officially-stopped-hiring-drivers-in

[23] NYC covid-19 trends and totals. https://www.nyc.gov/site/doh/covid/covid-19-data-totals.page.

A Appendix A

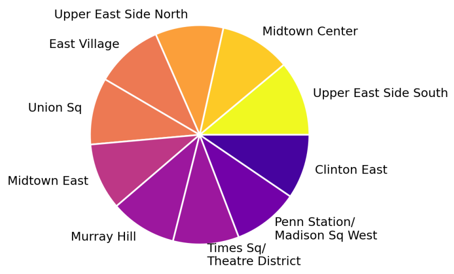


Figure 15: Top 10 Pick up locations 2015

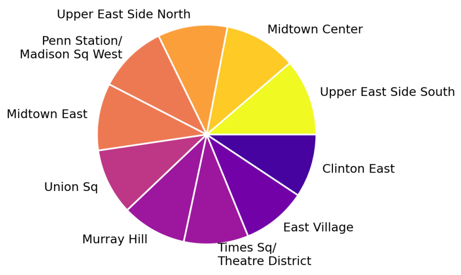


Figure 16: Top 10 Pick up locations 2016

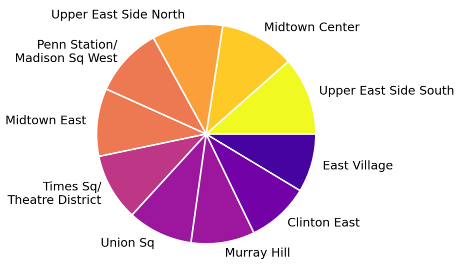


Figure 17: Top 10 Pick up locations 2017

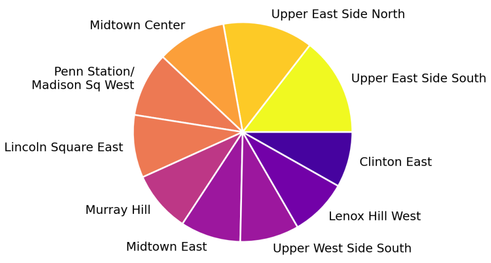


Figure 21: Top 10 Pick up locations 2021

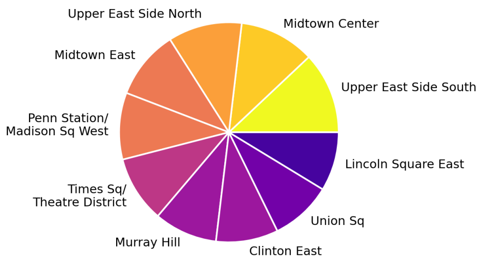


Figure 18: Top 10 Pick up locations 2018

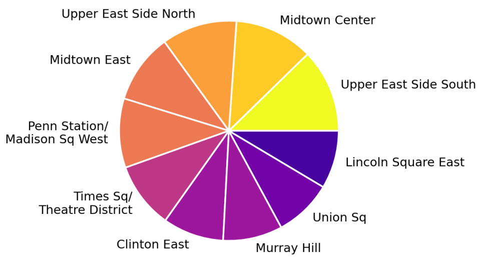


Figure 19: Top 10 Pick up locations 2019

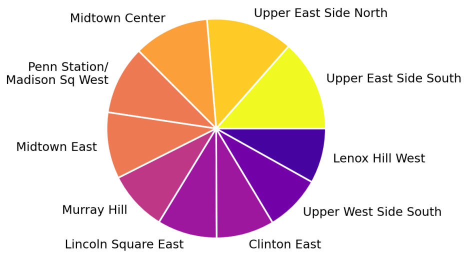


Figure 20: Top 10 Pick up locations 2020