Data Engineering - The New York Cabs

Rohit Bewoor (11011831)
Sanika Medankar (11011861)
Tushar Dwivedi (11011834)

SRH HOCHSCHULE HEIDELBERG
Intelligence in Learning

INTRODUCTION

STAATLICH ANERKANNTE HOCHSCHULE

- Project to cover data ingestion, storage and analysis (visualisation) preferably using Kafka for publish-subscribe modelling

- Data Source is a CSV file available on New York Taxi data repository site
  - Monthly data file for three types of taxis are available
  - Each row represents one trip
  - Due to size and limited processing power
    - project uses only Yellow Taxi June 2018 file.
    - Analysis on 28 days (from 2018-06-01 00:00:00 to 2018-06-29 00:00:00) based on PickUp time
    - June CSV file stats: 750 mb, approx. 8.5 million rows, 17 fields per row

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
1	01-06-2018 00:15	01-06-2018 00:16	1	0	1	N	145	145	2	3	0.5	0.5	0	0	0.3	4.3

- Data stored routed via Kafka and then stored in NoSQL database.

Tools

STAATLICH
ANERKANNTE
HOCHSCHULE

- Kafka as a Publisher-subscriber streaming platform:
  - To stream raw data from CSV file.
  - Subscribe and process before storing in NoSQL database.

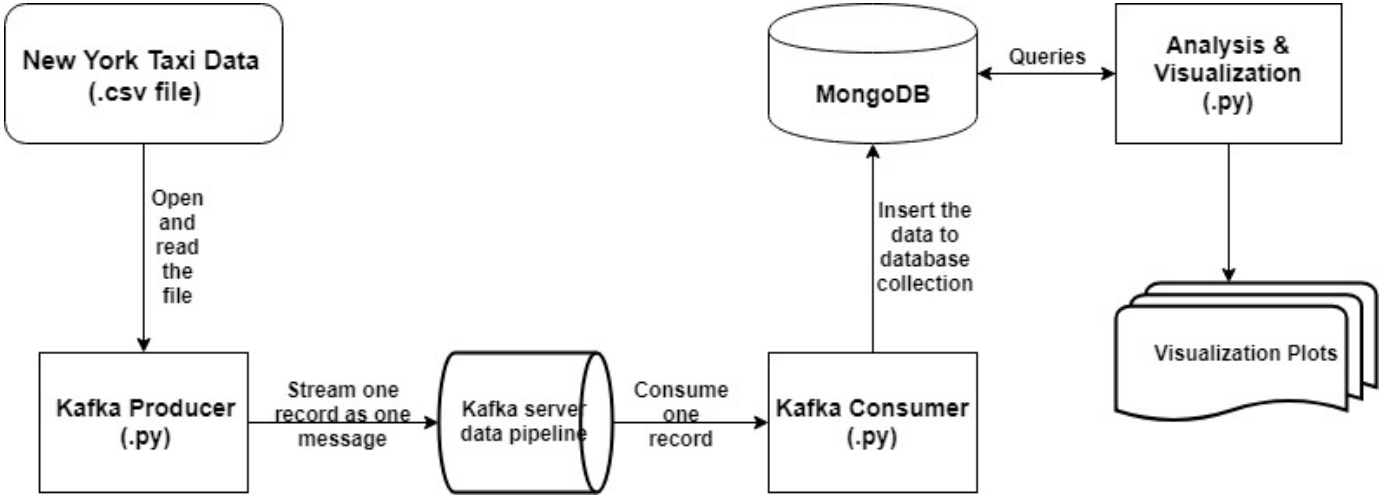
- MongoDB:
  - Holds data for later querying and analysis

- Python Language:
  - Jupyter Notebook and Spyder: All scripts for producer, consumer and analysis developed using Python.
  - Packages used were Pymongo, Pandas, Numpy, Kafka, Matplotlib, Json

Pipeline Structure

Kafka Consumer could be run as single or as multiple instances using group_id

STAATLICH ANERKANNTE HOCHSCHULE



Kafka Producer

STAATLICH ANERKANNTE HOCHSCHULE

- Topic with 5 partitions
- Open the CSV file in read format
- Parse the file and keep streaming one row data at a time as a message
- Perform boundary check for PUdt being within June 1st to 30th dates before sending

```
22 boundLowerPUDatetime = '2018-06-01 00:00:00'    # the pickup datetime from csv to be greater or equal to this
23 boundUpperPUDatetime = '2018-07-01 00:00:00'    # the pickup datetime from csv to be lower than this
```

```
Sent 8500439 row as the 8500000 th message to Kafka
Processing csv row number 8600000 at  Thu Jan 10 00:25:43 2019
Processing csv row number 8700000 at  Thu Jan 10 00:26:45 2019


Processed 8713832 rows from csv
Sent 8713307 messages to Kafka
NOT SENT 524 rows as OB

EndTime is: Thu Jan 10 00:26:53 2019
```

```
if (row_tpep_pickup_datetime < boundUpperPUDatetime and row_tpep_pickup_datetime >= boundLowerPUDatetime):
    joinedRow = ','.join(row)
    producer.send(topicName, joinedRow.encode('utf-8'))
    countSent2Producer = countSent2Producer + 1
```

```
C:\Everything\Kafka2.1.0-Scala2.12\bin\windows>kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 5 --topic TestNYTJune30days
Created topic "TestNYTJune30days".
```

Kafka Consumer

STAATLICH ANERKANNTE HOCHSCHULE

- Kafka consumer executed with group id, auto commit enabled, commit interval of 2 seconds, auto offset reset as ‘Earliest’
- Connect the kafka consumer with the Mongodb client
- Consume the data in kafka pipeline, create the JSON and insert into MongoDb collect
  - PUdt with first 28 days data written to collection #1
  - Remaining 2 days messages routed to collection #2

```
consumer = KafkaConsumer(
    bootstrap_servers=['localhost:9092'],
    auto_offset_reset='earliest',
    enable_auto_commit=True,
    auto_commit_interval_ms=2000,
    group_id='SingleRunning1')
#     consumer_timeout_ms=5000)
```

```
C:\Everything\Kafka2.1.0-Scala2.12\bin\windows>kafka-consumer-groups.bat --bootstrap-server localhost:9092 --group MultipleRunning30-1 --describe

TOPIC           PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG             CONSUMER-ID                                     HOST            CLIENT-ID
TestNYTJune30days 1          11761           90762           79001           kafka-python-1.4.4-1c2398d1-82de-4b69-9d4f-921c888e3662 /172.19.182.0   kafka-python-1.4.4
TestNYTJune30days 2          13772           89991           76219           kafka-python-1.4.4-d3519efc-c2f1-4462-8074-647d6515d0da /172.19.182.0   kafka-python-1.4.4
TestNYTJune30days 3          18786           90151           71365           kafka-python-1.4.4-d9b6dbd1-bee1-44ae-a215-b01918cdaee7 /172.19.182.0   kafka-python-1.4.4
TestNYTJune30days 0          12736           90369           77633           kafka-python-1.4.4-08e3a573-8e61-4e7c-8ba4-a19a2bfad366 /172.19.182.0   kafka-python-1.4.4
TestNYTJune30days 4          12726           90631           77905           kafka-python-1.4.4-da815fbb-9da4-4144-b007-1a1268948001 /172.19.182.0   kafka-python-1.4.4
```

- Single v/s Multi consumer runs
  - Run time slower for Single
  - For multi-run, broker at times did not utilise all consumers and rebalanced load to assign multiple partitions to same consumer. Often had only 3 and later only 2 running.

# of consumers in parallel	Runtime	Remarks
5	82 mins	Same computer (hardware Win10 with 8Gb ram). Producer and Multiple Consumers running with group name MultipleRunning30-1 at same time. At times, some consumers have already processed messages in the partition and are idle. Single instance consumer run after Producer ended already. So it ran without any downtime unlike some consumers in the multi-instance scenario. Group name SingleRunning30-1 Approx. 8.7 million messages consumed.
1	115 mins	

Analysis

STAATLICH ANERKANNTE HOCHSCHULE

- Python script execute queries to MongoDb and perform data building up of calculated fields
- Plot the query result using the python libraries i.e. matplotlib

```
Analysis data:


Total Trips per day:
Day1 = 320529.0000 -- Day2 = 308393.0000 -- Day3 = 265244.0000 -- Day4 = 279353.0000 -- Day5 = 298140.0000 -- Day6 = 315354.00
00 -- Day7 = 319315.0000 -- Day8 = 315660.0000 -- Day9 = 293934.0000 -- Day10 = 249740.0000 -- Day11 = 278488.0000 -- Day12 =
295377.0000 -- Day13 = 312651.0000 -- Day14 = 316168.0000 -- Day15 = 281883.0000 -- Day16 = 275862.0000 -- Day17 = 247840.0000
-- Day18 = 281485.0000 -- Day19 = 298528.0000 -- Day20 = 307880.0000 -- Day21 = 309069.0000 -- Day22 = 311362.0000 -- Day23 =
273667.0000 -- Day24 = 243786.0000 -- Day25 = 267868.0000 -- Day26 = 287191.0000 -- Day27 = 297916.0000 -- Day28 = 302796.0000
--

Total Passenger per day:
Day1 = 507899.0000 -- Day2 = 509665.0000 -- Day3 = 431153.0000 -- Day4 = 436819.0000 -- Day5 = 466506.0000 -- Day6 = 494194.00
00 -- Day7 = 499014.0000 -- Day8 = 501788.0000 -- Day9 = 485392.0000 -- Day10 = 408601.0000 -- Day11 = 436463.0000 -- Day12 =
463919.0000 -- Day13 = 491022.0000 -- Day14 = 497015.0000 -- Day15 = 438738.0000 -- Day16 = 453013.0000 -- Day17 = 411547.0000
-- Day18 = 444826.0000 -- Day19 = 470030.0000 -- Day20 = 487199.0000 -- Day21 = 487827.0000 -- Day22 = 499181.0000 -- Day23 =
455177.0000 -- Day24 = 402607.0000 -- Day25 = 425725.0000 -- Day26 = 455372.0000 -- Day27 = 469436.0000 -- Day28 = 479499.0000
--

Average Passerger/Trip per day:
Day1 = 1.5846 -- Day2 = 1.6526 -- Day3 = 1.6255 -- Day4 = 1.5637 -- Day5 = 1.5647 -- Day6 = 1.5671 -- Day7 = 1.5628 -- Day8 =
1.5896 -- Day9 = 1.6514 -- Day10 = 1.6361 -- Day11 = 1.5673 -- Day12 = 1.5706 -- Day13 = 1.5705 -- Day14 = 1.5720 -- Day15 =
1.5565 -- Day16 = 1.6422 -- Day17 = 1.6605 -- Day18 = 1.5803 -- Day19 = 1.5745 -- Day20 = 1.5824 -- Day21 = 1.5784 -- Day22 =
1.6032 -- Day23 = 1.6633 -- Day24 = 1.6515 -- Day25 = 1.5893 -- Day26 = 1.5856 -- Day27 = 1.5757 -- Day28 = 1.5836 --
```

From the Matrix data structure

```
The location list AFTER populating from db::

Day 1
D1  --  2018-06-01 00:00:00  --  2018-06-02 00:00:00
[['TopPULoc1', 'PULoc1-UNFILLED', 0], ['TopPULoc2', 'PULoc2-UNFILLED', 0], ['TopPULoc3', 'PULoc3-UNFILLED', 0], ['TopPULoc4',
'PULoc4-UNFILLED', 0], ['TopPULoc5', 'PULoc5-UNFILLED', 0]]
[['TopDOLoc1', 'DOLoc1-UNFILLED', 0], ['TopDOLoc2', 'DOLoc1-UNFILLED', 0], ['TopDOLoc3', 'DOLoc3-UNFILLED', 0], ['TopDOLoc4',
'DOLoc4-UNFILLED', 0], ['TopDOLoc5', 'DOLoc5-UNFILLED', 0]]
[['TopPUDOCombo1', '264 TO 264', 3276], ['TopPUDOCombo2', '237 TO 236', 2090], ['TopPUDOCombo3', '236 TO 236', 1878], ['TopPUD
OCombo4', '236 TO 237', 1764], ['TopPUDOCombo5', '237 TO 237', 1471], ['TopPUDOCombo6', '239 TO 238', 914], ['TopPUDOCombo7',
'239 TO 142', 908], ['TopPUDOCombo8', '237 TO 162', 859], ['TopPUDOCombo9', '238 TO 239', 814], ['TopPUDOCombo10', '141 TO 23
6', 814]]
```

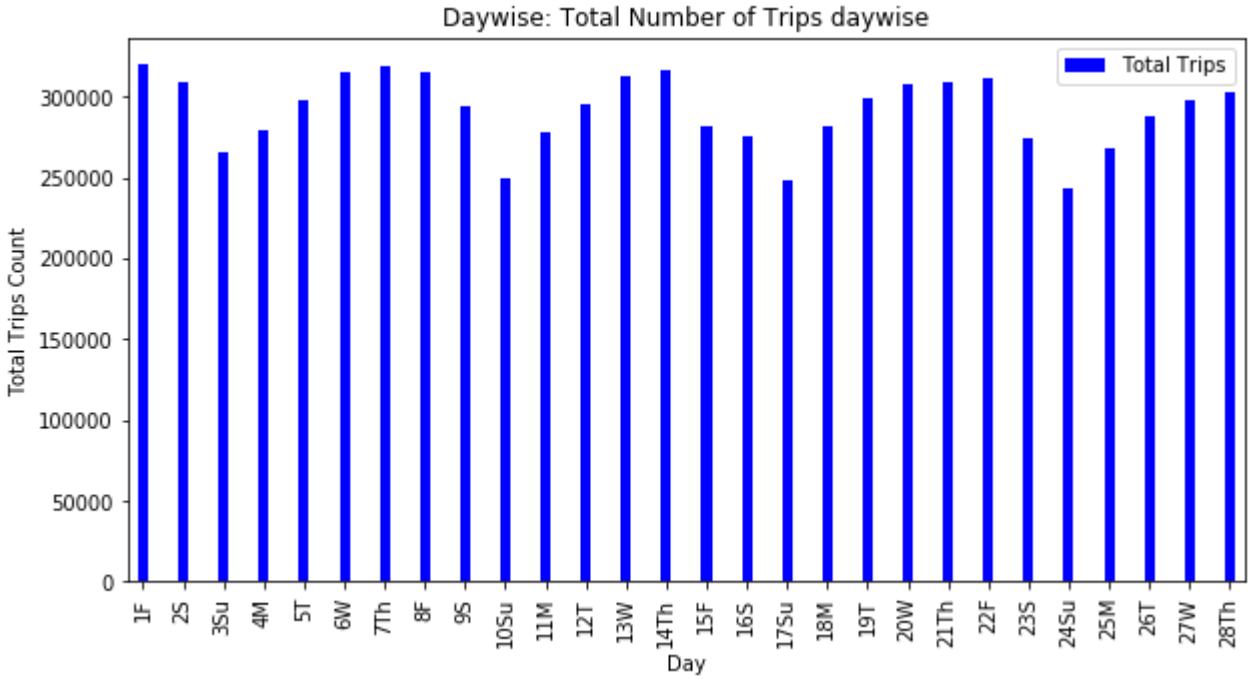
From the List data structure

Total Trips per day (28 days)

STAATLICH
ANERKANNTE
HOCHSCHULE

Day Wise Trips - 28 days

- Total trips vary from around 265k to 315k
- Maximum demand on Thursdays and Fridays, lowest on Sunday

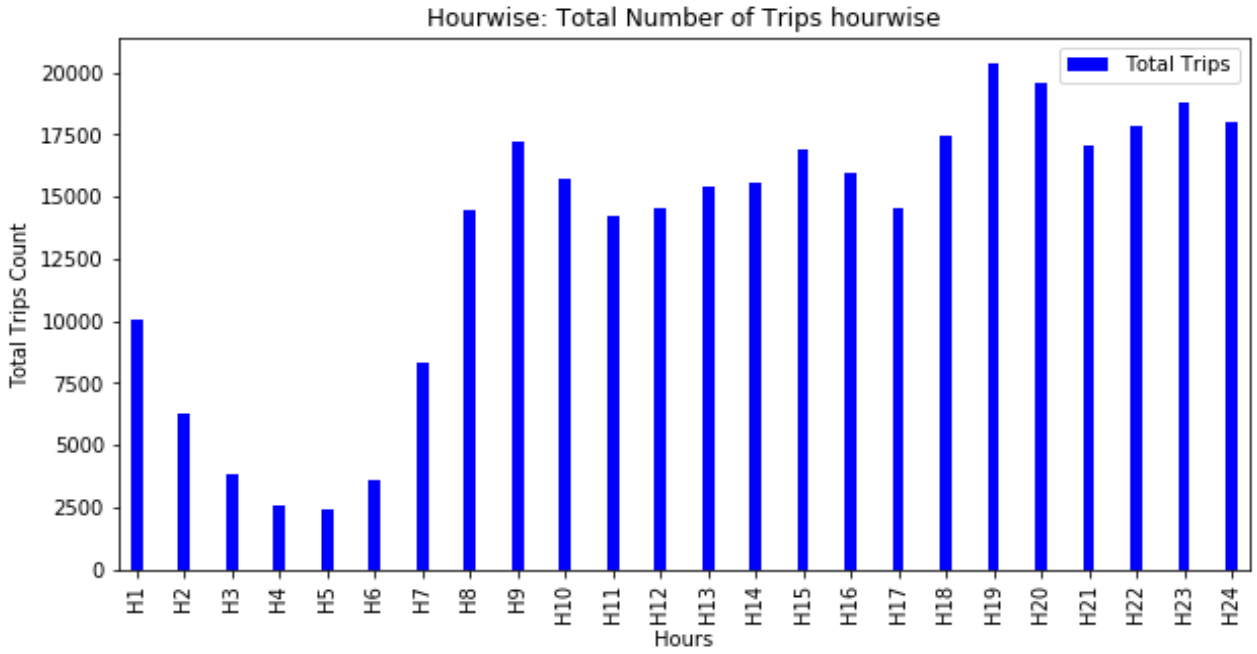


Total Trips per Hour (01June)

STAATLICH ANERKANNTE HOCHSCHULE

Hourwise Trips 01.06 Friday

- H2 to H6: Low number of trips during dead of night to slow uptick between 0500 to 0600
- H8 to H18: Number of trips almost steady around 16000 per hour
- H19 to H24: Number of trips rises to 20k+ and remains higher than before H18 - probably due to Friday night and late turn in by NY population

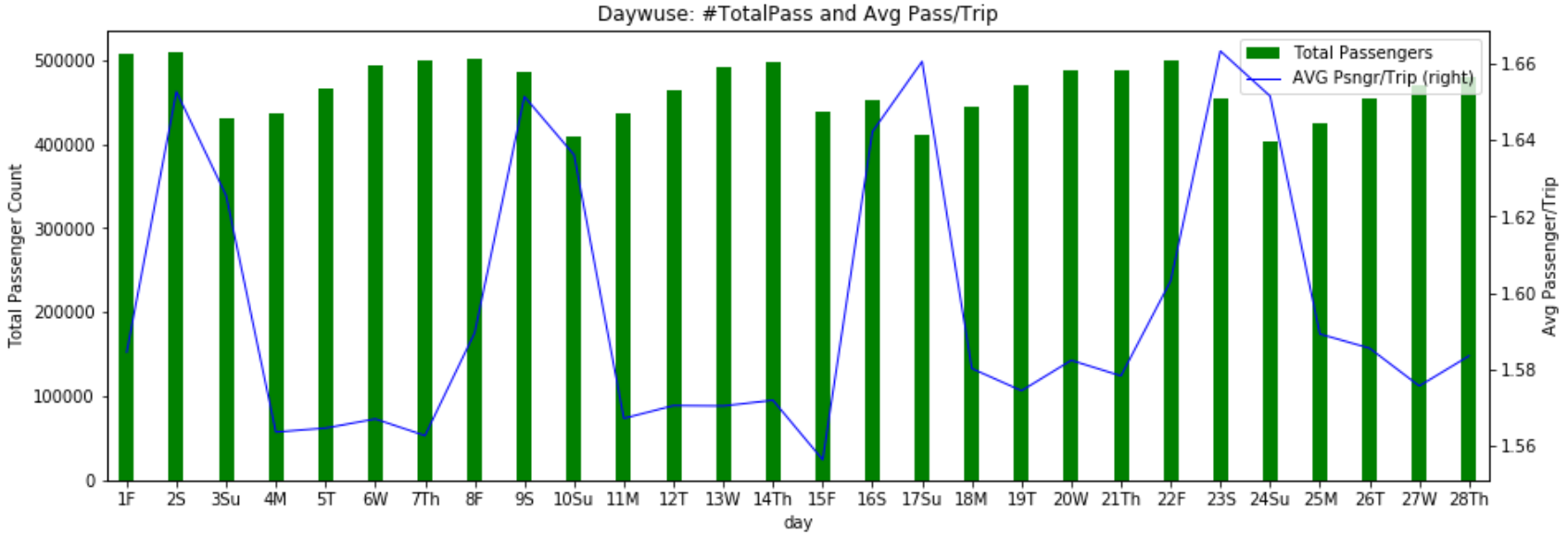


Total Passengers vs Average Passengers (28 days)

STAATLICH ANERKANNTE HOCHSCHULE

Day Wise variation 28 days

- Average Passengers per Trip low from Monday to Thursday and higher on weekends
- Single rider trips are less popular on weekends and shared trips are common
- Total number of passengers taking taxis follows the total trips trend cycle in general

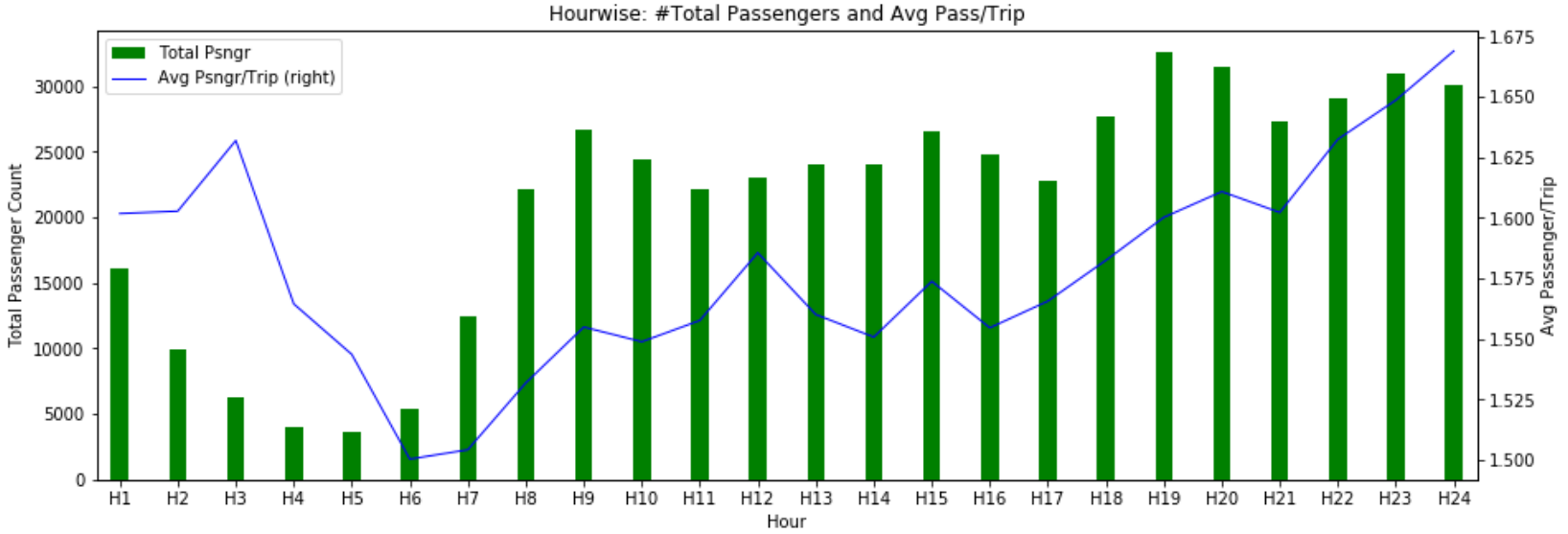


Total Passengers vs Average Passengers Hourly (1Jun)

STAATLICH ANERKANNTE HOCHSCHULE

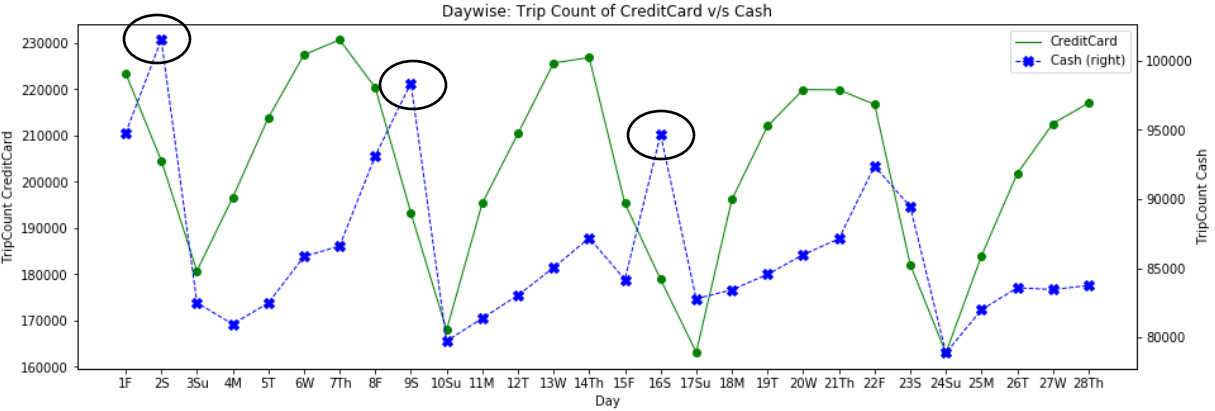
Hourwise variation on 01.06 Friday

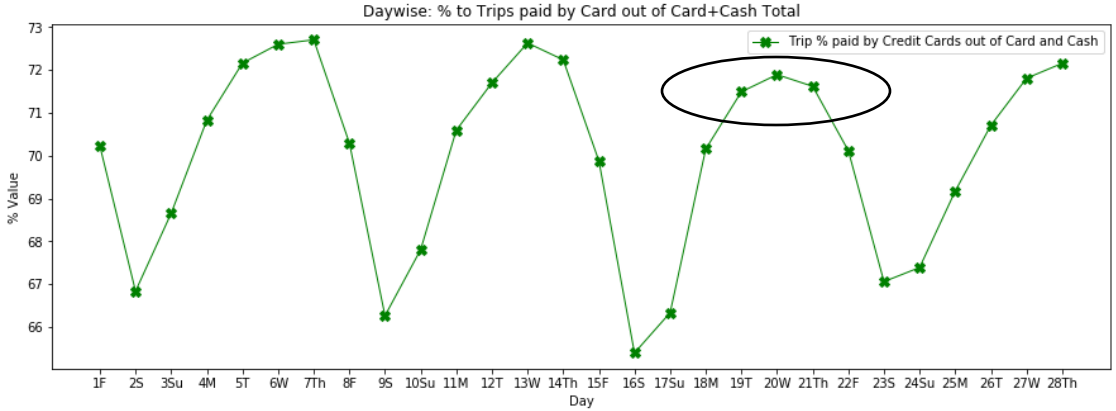
- H3 to H5: Tendency for multi passenger trips is higher
  => Total passengers dropped by 40% (6293 to 3677), (total number of trips fell by 40%), Avg Psng/Trip fell 5% (1.63 to 1.54)
- H12 to H14: Single ride tendency increased
  Passenger total remained steady or increased very slightly, but the avg Psng/Trip fell by about 3%



Credit Card vs Cash payment method- Daily (28 days)

STAATLICH ANERKANNTE HOCHSCHULE



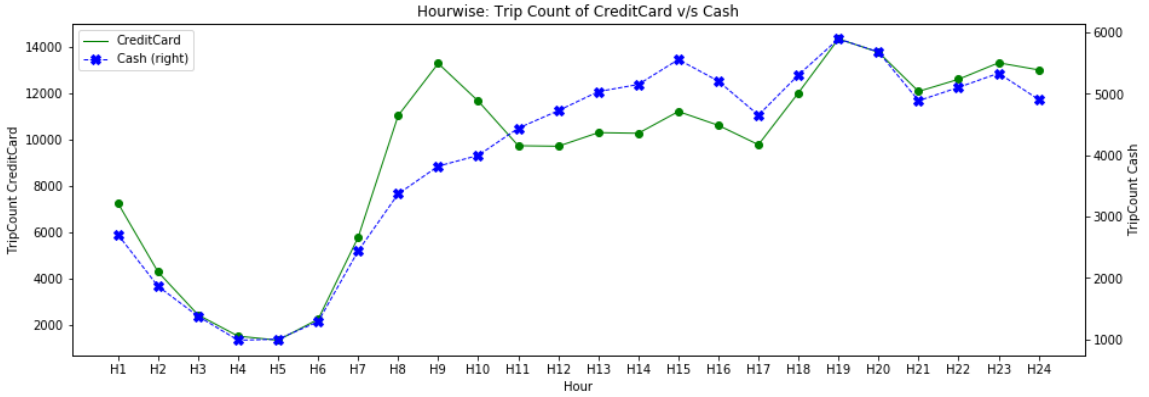


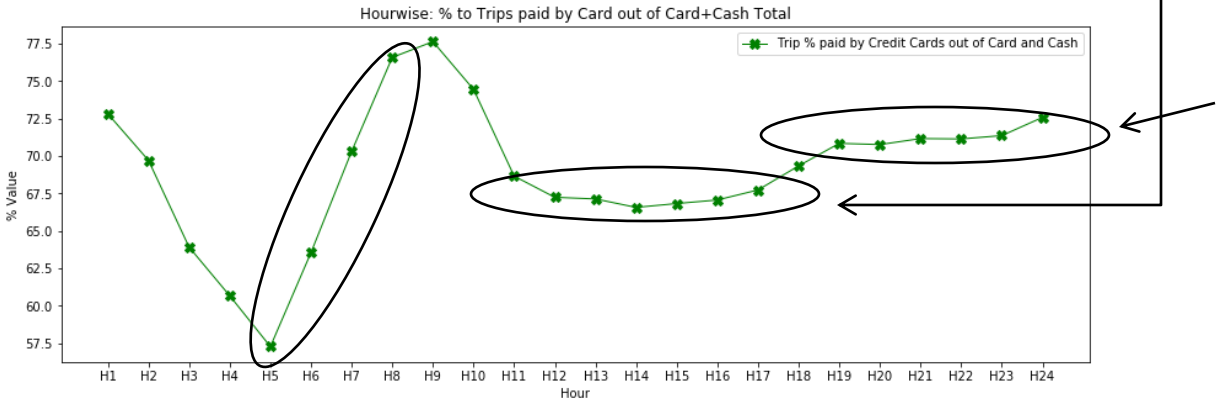
Daily variation 28 days

- Share of credit card payments as against cash
  - Maximum - Tuesdays to Thursdays.
  - Lowest on weekends
- 3 out 4 Saturdays rank highest in the week for cash paid trips

Credit Card vs Cash payment method – Hourly (1Jun)

STAATLICH ANERKANNTE HOCHSCHULE





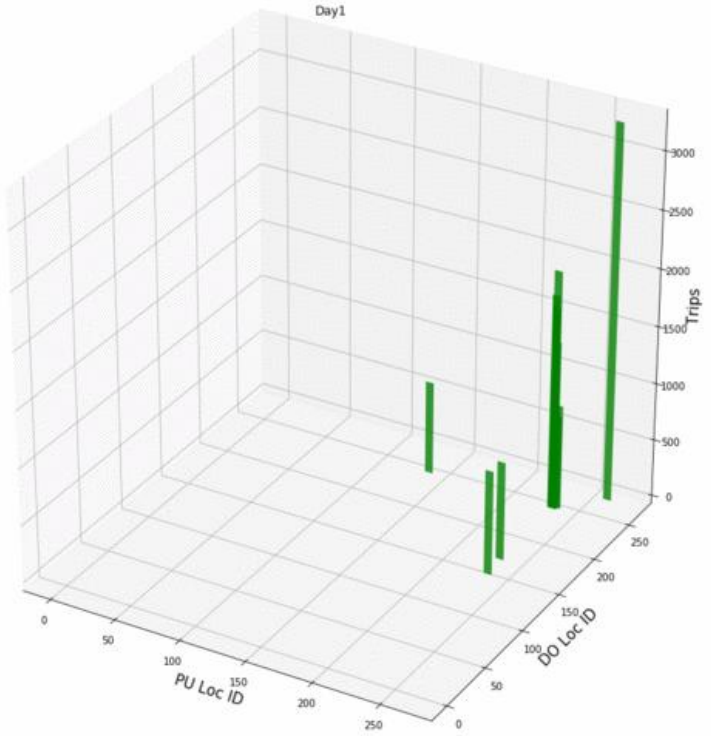
Hourly variation on 01.06 Friday

- H5 to H9: Ratio of Card payment against trips paid for by only Cash i.e Card payment continued to increase steadily from approx 60% to 80%
- H9 to H17: The overall use dropped to about 65% by H12 and remained steady till H17
- H17 to H24: Increased slightly till H19 and remained steady at approx. 72%

Top 10 PU to DO variation – 28 days

Fr	Sa	Su	Mo	Tu	We	Th
1,8, 15,22	2,9, 16,23	3,10, 17,24	4,11, 18,25	5,12, 19,26	6,13, 20,27	7,14, 21,28

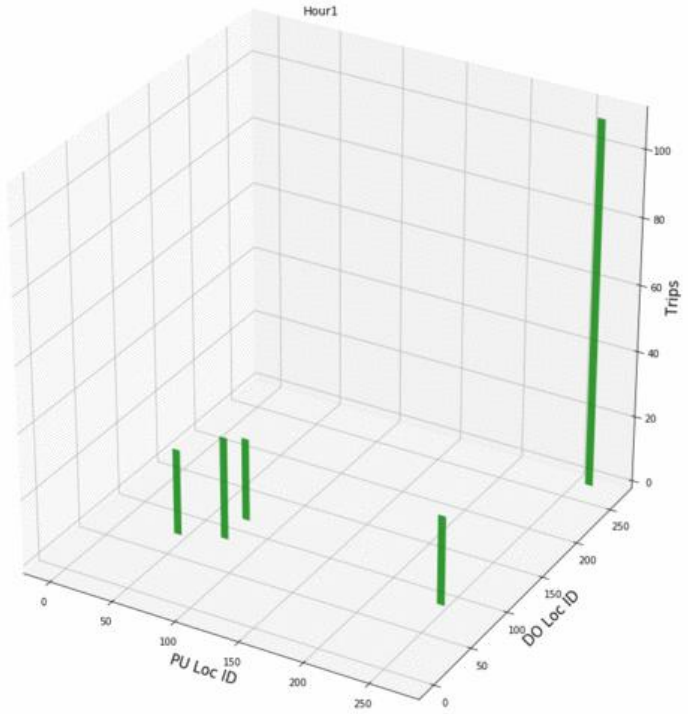
- PU Loc 264 to DO Loc 264 appears within top 10 each day and it has the highest demand daily with over 3000 trips per day
- Trips to and fro for combinations of LocID's 236 and 237 feature consistently in the next top 4 routes by demand.
- Except on Saturday's and Sunday's, all the top 10 locations ID's of PU to DO combo are concentrated around the 200 locations.



Top 5 PU to DO route variation – hourly (1Jun)

Hourly Variation of PU-DO route on 01.06

- H1 to H22: The top 5 PU to DO locations are almost the same throughout
- Location 264: PU Loc 264 to DO Loc 264 appears in 23 out of the 24 hours within top 5 list.
  Also, for 17 of these hours it has the highest number of trips.
- Variability in trip count for top most PU-DO routes is from low 20's (H4 to H5) and peaks around 225 range (H22 to H24)



0000-0100 hrs

PULoc 48 to DOLoc 68 = 26 trips
PULoc 79 to DOLoc 79 = 31 trips
PULoc 249 to DOLoc 79 = 27 trips
PULoc 79 to DOLoc 107 = 25 trips
PULoc 264 to DOLoc 264 = 110 trips

0100-0200 hrs

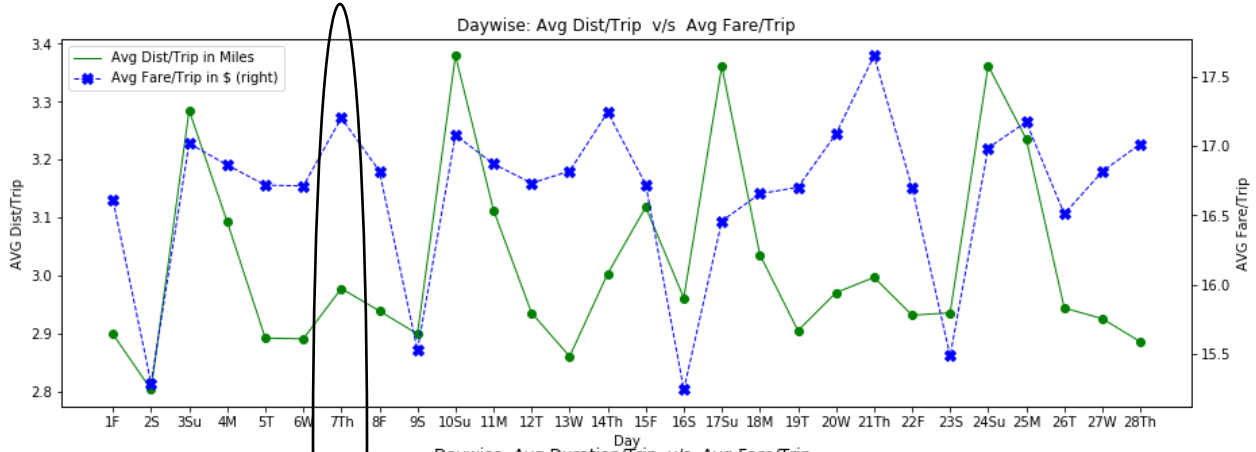
PULoc 48 to DOLoc 48 = 23 trips
PULoc 79 to DOLoc 79 = 27 trips
PULoc 114 to DOLoc 79 = 22 trips
PULoc 148 to DOLoc 79 = 26 trips
PULoc 264 to DOLoc 264 = 69 trips

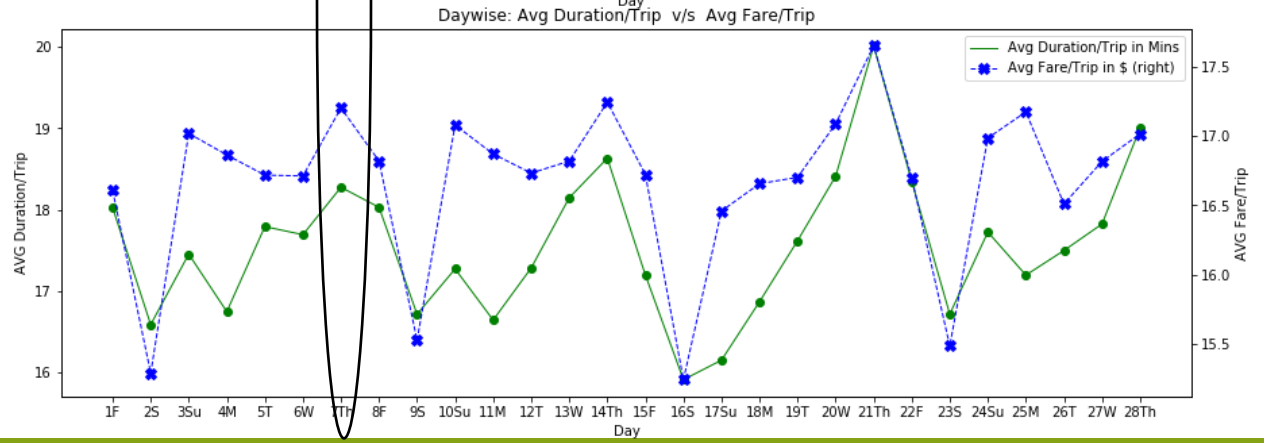
2300-0000 hrs

PULoc 48 to DOLoc 48 = 54 trips
PULoc 48 to DOLoc 68 = 60 trips
PULoc 79 to DOLoc 79 = 50 trips
PULoc 79 to DOLoc 148 = 52 trips
PULoc 264 to DOLoc 264 = 228 trips

Day-Wise Avg Fare v/s Duration per Trip – 28 days

STAATLICH ANERKANNTE HOCHSCHULE



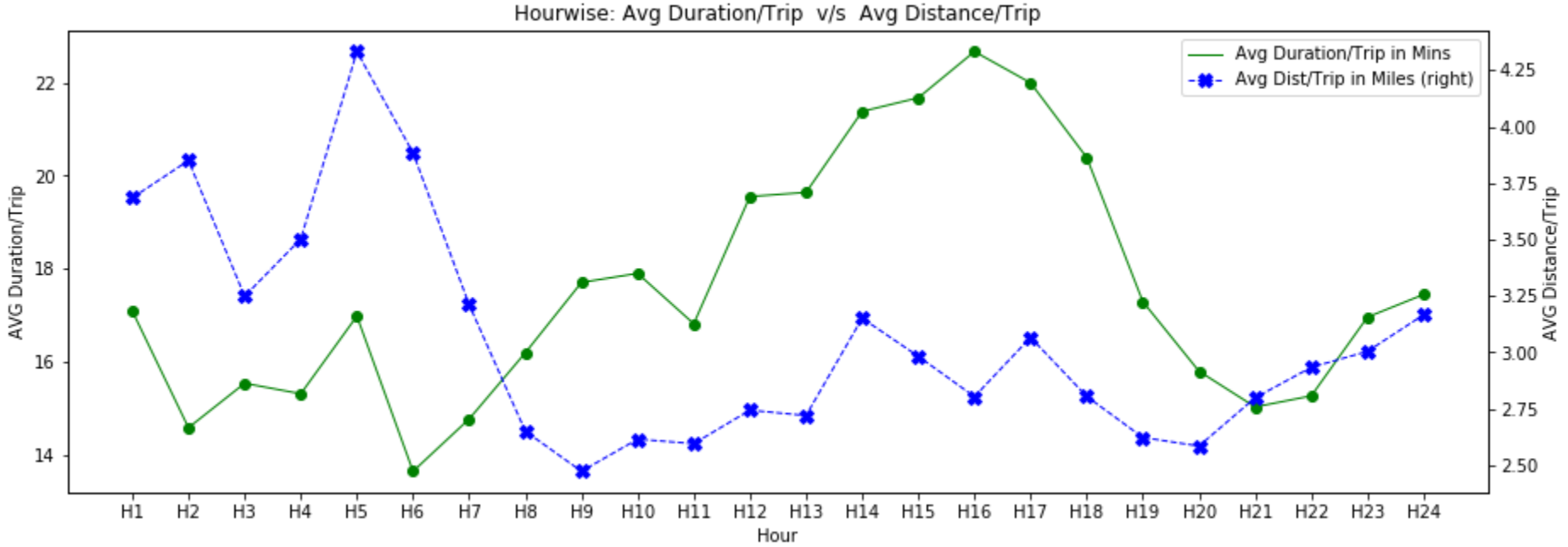


- Avg Fare/trip evinces clear pattern in first two weeks of June with troughs on Saturday, peaks on Sunday and Thursday
- Avg Distance is consistently high on Sundays and that impacts the Avg fare making it high even though the Avg Duration is on lower side
- On Thursday the high Avg Duration has same effect and low Avg Distance has limited impact
- On 17th as duration remained low the regular peak on Sunday was subdued even with high Avg Dist
- On 21st Thur the fare is really high as duration goes higher than normal

Avg Distance and Duration per Trip – Hourwise (1Jun)

STAATLICH ANERKANNTE HOCHSCHULE

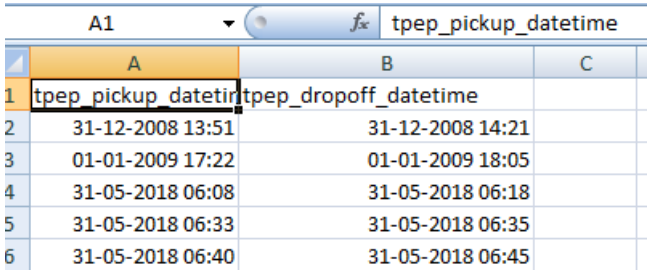
- During early hours (H2 to H7) duration per trip remains low while the distance per trip remains high - supports low congestion during dead of night.
- H6 to H17, congestion picks up as distance per trip falls to a low steady value but duration per trip continues to increase.
- H17 onwards duration keeps falling as distance falls again.



Challenges and Learnings

STAATLICH ANERKANNTE HOCHSCHULE

1) The CSV file had over 8 million rows and opening it in Excel truncated and showed only around first one million rows. Sorting on the PUdt showed there were records with dates of May 2018 and a few stray records from 2008 and 2009 too. Therefore, we are implementing a programmatic check in the producer to read the whole CSV file.



2) Kafka consumer parameters: We were unable to use either the synchronous or an asynchronous manual commit issued from within the program at a logical point - ideally on return to program after completion of successful MongoDB insert. Therefore, we are relying purely on auto-commit being enabled.

```
consumer = KafkaConsumer(
    bootstrap_servers=['localhost:9092'],
    auto_offset_reset='earliest',
#     auto_offset_reset='latest',    # default is latest
    enable_auto_commit=True,
    auto_commit_interval_ms=2000,
    group_id='SingleRunning1')
```

Challenges and Learnings

STAATLICH
ANERKANNTE
HOCHSCHULE

3) During testing we found that if the trip distance was 0 then our consumer program picked up the field value as .00 and created an error while building the JSON for MongoDb write. So special code was inserted to handle this situation..

```
52        msgAsList = msgAsString.split(",")
53        trip_distanceAsString = str(msgAsList[4])
54        if trip_distanceAsString[0:1] == '.':        # a 0 value comes as  .00 so checking and making as 0.00
55            trip_distanceAsString = '0' + trip_distanceAsString
```

4) An attempt was made to analyse the data streamed in kafka using spark. There are some hurdles for integrating the kafka server and spark. Even after spark installations were completed there were issues with the spark initializations.
We wanted to show a real time analysis to display :

- The trip with highest fare streamed thus far
- The trip with highest distance.
- The running count of the trips originating with a certain pick up location (PU) id eg. Location id 264 has 80 trips so far.

Since, we were unable to start Spark, we could not perform the above mentioned analysis.

Bibliography

STAATLICH
ANERKANNTE
HOCHSCHULE

1) NYC Taxi related site
http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml
http://www.nyc.gov/html/tlc/html/home/home.shtml
2) Data dictionary for Yellow Taxi file
http://www.nyc.gov/html/tlc/downloads/pdf/data_dictionary_trip_records_yellow.pdf
3) Kafka and Zookeeper Related Sites
https://towardsdatascience.com/kafka-python-explained-in-10-lines-of-code-800e3e07dad1
https://www.programcreek.com/python/example/98440/kafka.KafkaConsumer
https://blog.workwell.io/how-to-manage-your-kafka-consumers-from-the-producer-9933b88085dd
https://medium.com/@Ankitthakur/apache-kafka-installation-on-mac-using-homebrew-a367cdefd273
4) Mongo Related Sites
https://stackoverflow.com/questions/17053323/how-to-order-mongodb-aggregation-with-match-sort-and-limit
http://www.forwardadvance.com/course/mongo/mongo-aggregation/aggregation-count
https://docs.mongodb.com/manual/reference/operator/aggregation/dateFromString/
https://stackoverflow.com/questions/41564510/convert-string-date-to-timestamp-in-mongodb
5) Python related
https://realpython.com/python-csv/
https://realpython.com/working-with-large-excel-files-in-pandas/
http://zetcode.com/python/pymongo/
https://www.geeksforgeeks.org/python-string-split/
https://api.mongodb.com/python/current/api/pymongo/mongo_client.html#pymongo.mongo_client.MongoClient
https://www.youtube.com/watch?v=WX0MDddgpA4
https://www.youtube.com/watch?v=w9tAoscq3C4
6) Spark Implementation
https://medium.com/@GalarnykMichael/install-spark-on-windows-pyspark-4498a5d8d66c
https://medium.com/@mukeshkumar_46704/getting-streaming-data-from-kafka-with-spark-streaming-using-python-9cd0922fa904
https://spark.apache.org/downloads.html
https://www.youtube.com/watch?v=l8jMvfOU3vQ
http://spark.apache.org/docs/latest/building-spark.html#setting-up-mavens-memory-usage
7) Miscellaneous
https://gifmaker.me/