SSZG584 Data Management of IOT

Assignment 2
Data Processing with Apache Kafka

By :
Shashank Karrthikeyaa A S,
Roll Number : 2020MT12014
Group No : 7
E-Mail : 2020MT12014@wilp.bits-pilani.ac.in

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
ज्ञानं परमं बलम्

Contents

List of Tables **i**

List of Figures **ii**

1 Assignment 2 Data Processing with Apache Kafka **1**

1.1 Problem Statement 1
1.2 Architecture 1
1.3 Table Structure 2
1.4 Linux VM on AWS 2
1.5 Simulater Program for Truck Movement 3
1.6 Simulater to MQTT, MQTT to Kafka 6
1.7 Over Speeding Events 8
1.8 Data Summary 11
1.9 Conclusion 13

A Simulator Code for Truck 1 **14**

B Simulator Code for Truck 2 **19**

C Simulator Code for Truck 3 **24**

D Configuration for Trucks **29**

E SQL Queries **31**

F MQTT to Kafka and Raw Data **32**

G Kafka Overspeeding Producer **35**

H Kafka Overspeeding Consumer **38**

I Kafka Summary Producer **39**

J Kafka Summary Consumer **42**

List of Tables

1.1 Table Schema 2

List of Figures

1.1 Architecture . . . 1
1.2 SQL . . . 2
1.3 VM Instance State . . . 2
1.4 Firewall Inbound Rules . . . 3
1.5 Firewall Outbound Rules . . . 3
1.6 Config for Truck 1 . . . 4
1.7 Config for Truck 2 . . . 4
1.8 Config for Truck 3 . . . 5
1.9 Output for Truck 1 . . . 5
1.10 Output for Truck 2 . . . 6
1.11 Output for Truck 3 . . . 6
1.12 Data of Truck 1 and Truck 2 . . . 7
1.13 Data of Truck 3 . . . 7
1.14 Raw Data Starting . . . 8
1.15 Raw Data Ending . . . 8
1.16 Over Speeding Producer for Truck 1 . . . 9
1.17 Over Speeding Consumer for Truck 1 . . . 9
1.18 Over Speeding Producer for Truck 2 . . . 10
1.19 Over Speeding Consumer for Truck 2 . . . 10
1.20 Over Speeding Producer for Truck 3 . . . 11
1.21 Over Speeding Consumer for Truck 3 . . . 11
1.22 Data Summary Producer . . . 12
1.23 Data Summary Consumer . . . 13

Chapter 1

Assignment 2 Data Processing with Apache Kafka

1.1 Problem Statement

To prepare a prototype of Intelligent Vehicle Monitoring System using Open Source messaging platform Apache Kafka.

1.2 Architecture

The Architecture of the system is shown in the Fig. 1.1

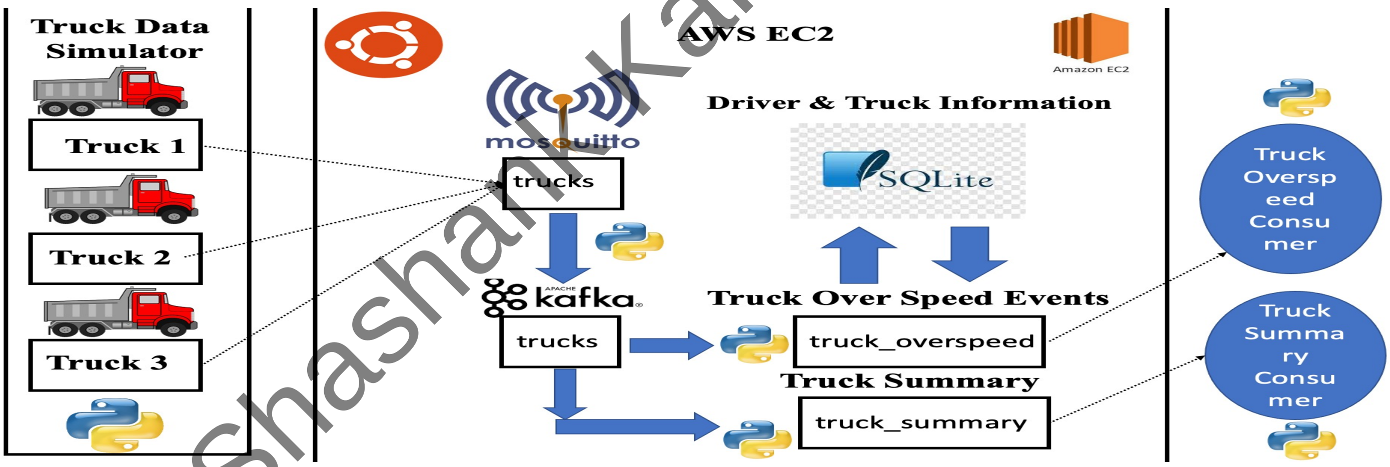


Figure 1.1: Architecture

The trucks send their location and speed via MQTT to the Mosquitto Broker on the AWS EC2 instance under the topic trucks. A python script subscribes to those topics and publishes the same messages on Apache Kafka under the topic trucks. A python script consumes this data from kafka and predicts overspeeding events. Currently Speed ¿ 1000 is considered as overspeeding events. Similarly a Python Script consumes the same data and summarizes for every 2 Minutes for all the trucks.

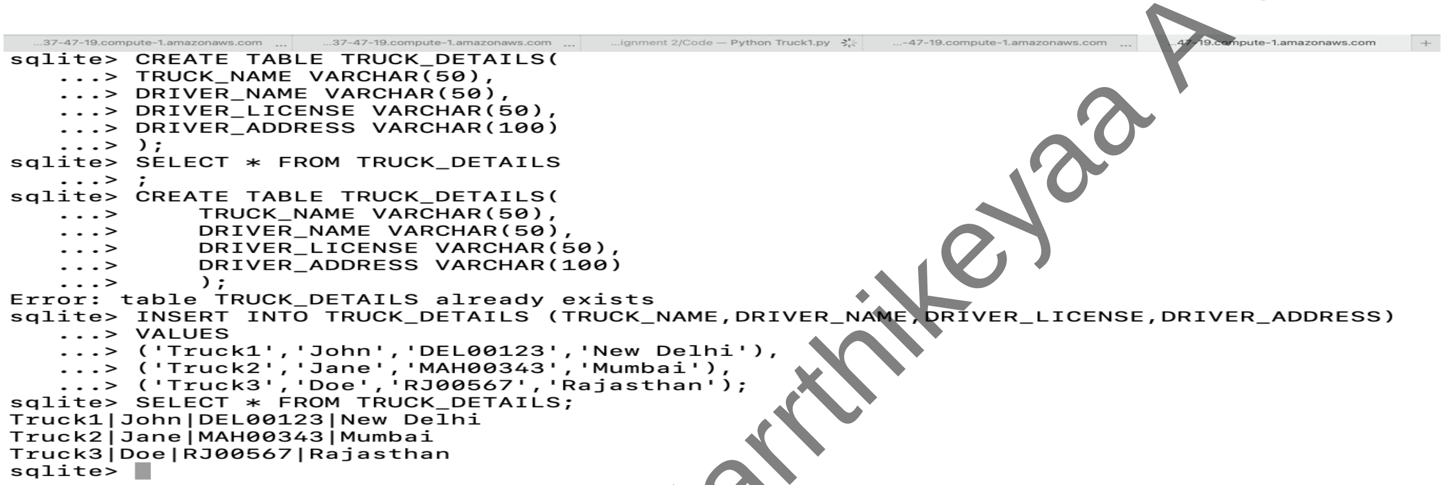
1.3 Table Structure

Table 1.1: Table Schema

COLUMNS	TRUCK_NAME	DRIVER_NAME	DRIVER_LICENSE	DRIVER_ADDRESS
DATA TYPE	VARCHAR(50)	VARCHAR(50)	VARCHAR(50)	VARCHAR(100)

The table structure followed can be found above in the Table 1.1. Truck Name , Driver Name, Driver License Details and Driver Address are stored.

The Create , Insert and Select SQL Statements can be found in the Figure 1.2



...37-47-19.compute-1.amazonaws.com
...37-47-19.compute-1.amazonaws.com
...ignment 2/Code — Python Truck1.py
...-47-19.compute-1.amazonaws.com
...47-19.compute-1.amazonaws.com
+

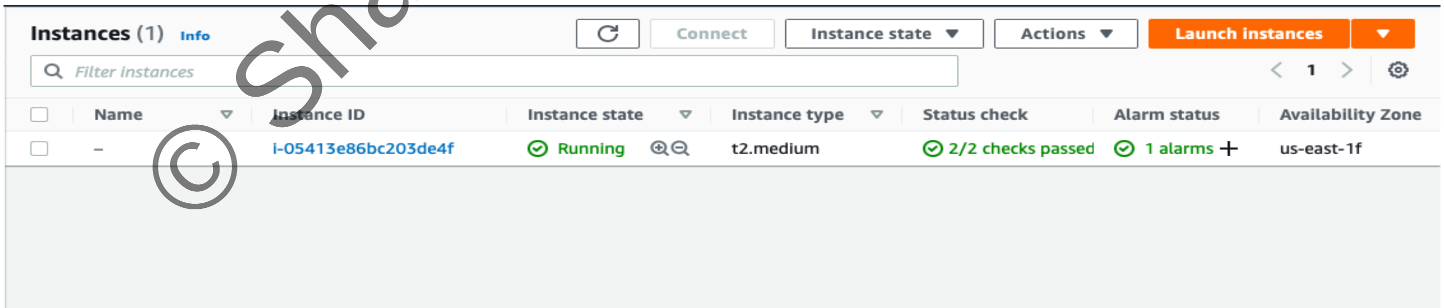
```
sqlite> CREATE TABLE TRUCK_DETAILS(
   ...> TRUCK_NAME VARCHAR(50),
   ...> DRIVER_NAME VARCHAR(50),
   ...> DRIVER_LICENSE VARCHAR(50),
   ...> DRIVER_ADDRESS VARCHAR(100)
   ...> );
sqlite> SELECT * FROM TRUCK_DETAILS
   ...> ;
sqlite> CREATE TABLE TRUCK_DETAILS(
   ...>     TRUCK_NAME VARCHAR(50),
   ...>     DRIVER_NAME VARCHAR(50),
   ...>     DRIVER_LICENSE VARCHAR(50),
   ...>     DRIVER_ADDRESS VARCHAR(100)
   ...>     );
Error: table TRUCK_DETAILS already exists
sqlite> INSERT INTO TRUCK_DETAILS (TRUCK_NAME,DRIVER_NAME,DRIVER_LICENSE,DRIVER_ADDRESS)
   ...> VALUES
   ...> ('Truck1','John','DEL00123','New Delhi'),
   ...> ('Truck2','Jane','MAH00343','Mumbai'),
   ...> ('Truck3','Doe','RJ00567','Rajasthan');
sqlite> SELECT * FROM TRUCK_DETAILS;
Truck1|John|DEL00123|New Delhi
Truck2|Jane|MAH00343|Mumbai
Truck3|Doe|RJ00567|Rajasthan
sqlite> 
```

Figure 1.2: SQL

1.4 Linux VM on AWS

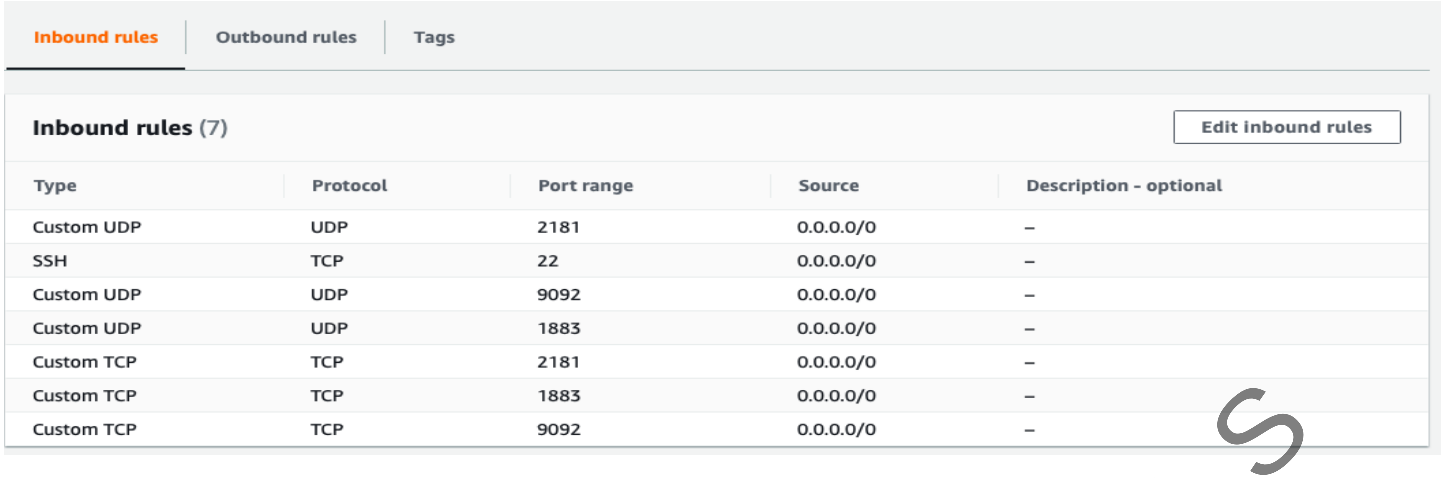
Ubuntu VM on AWS EC2 has been used to host this project.

The screenshot of the running instance can be found in the figure 1.3 and the firewall rules can be found in the figures 1.4 and 1.5



□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
□	–	i-05413e86bc203de4f	Running	t2.medium	2/2 checks passed	1 alarms +	us-east-1f

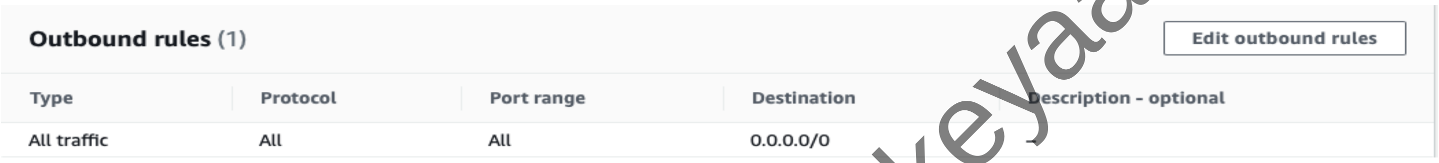
Figure 1.3: VM Instance State



Inbound rules (7)

Type	Protocol	Port range	Source	Description - optional
Custom UDP	UDP	2181	0.0.0.0/0	–
SSH	TCP	22	0.0.0.0/0	–
Custom UDP	UDP	9092	0.0.0.0/0	–
Custom UDP	UDP	1883	0.0.0.0/0	–
Custom TCP	TCP	2181	0.0.0.0/0	–
Custom TCP	TCP	1883	0.0.0.0/0	–
Custom TCP	TCP	9092	0.0.0.0/0	–

Figure 1.4: Firewall Inbound Rules



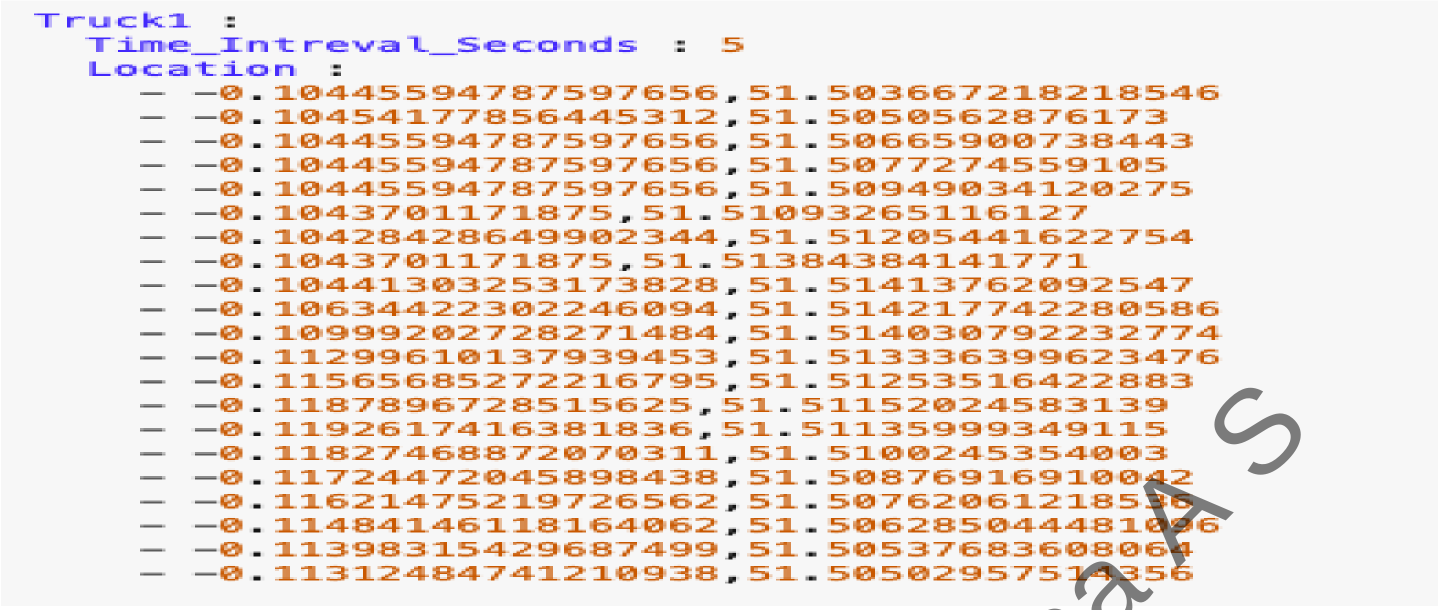
Outbound rules (1)

Type	Protocol	Port range	Destination	Description - optional
All traffic	All	All	0.0.0.0/0	–

Figure 1.5: Firewall Outbound Rules

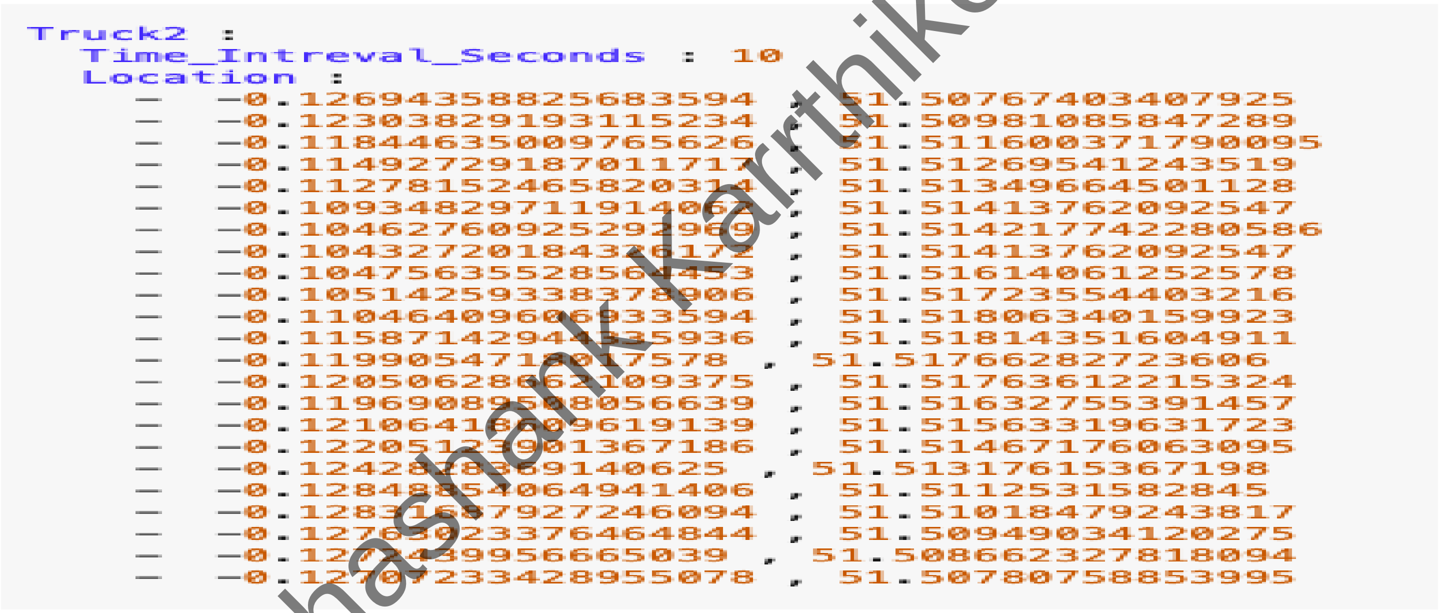
1.5 Simulater Program for Truck Movement

A Simulated Program is written using Python to simulate truck movements. The configuration for the trucks 1,2, and 3 can be found in the Figures 1.6,1.7,1.8 respectively. The outputs for the trucks 1,2 and 3 can be found in the Figures 1.9,1.10,1.11 respectively.
As you can see in the figures 1.9,1.10,1.11 each Truck will send it's ID, Time Stamp when the data is being sent , Latitude and Longitude of it's location and it's Speed.



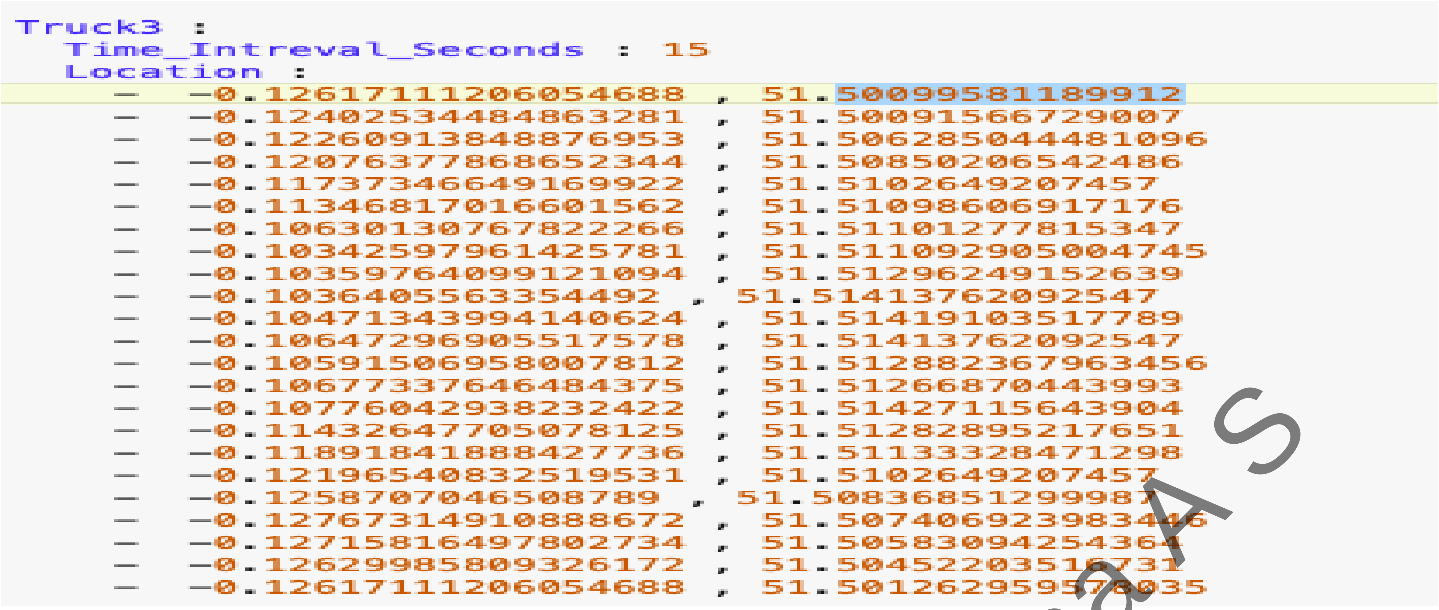
```
Truck1 :
  Time_Intreval_Seconds : 5
  Location :
    - -0.10445594787597656,51.503667218218546
    - -0.10454177856445312,51.5050562876173
    - -0.10445594787597656,51.50665900738443
    - -0.10445594787597656,51.5077274559105
    - -0.10445594787597656,51.50949034120275
    - -0.1043701171875,51.51093265116127
    - -0.10428428649902344,51.51205441622754
    - -0.1043701171875,51.51384384141771
    - -0.10441303253173828,51.51413762092547
    - -0.10634422302246094,51.514217742280586
    - -0.10999202728271484,51.514030792232774
    - -0.11299610137939453,51.513336399623476
    - -0.11565685272216795,51.51253516422883
    - -0.1187896728515625,51.51152024583139
    - -0.11926174163818836,51.51135999349115
    - -0.11827468872070311,51.51002453354003
    - -0.11724472045898438,51.50876916910042
    - -0.11621475219726562,51.50762061218536
    - -0.11484146118164062,51.506285044481096
    - -0.11398315429687499,51.50537683608064
    - -0.11312484741210938,51.50502957514356
```

Figure 1.6: Config for Truck 1



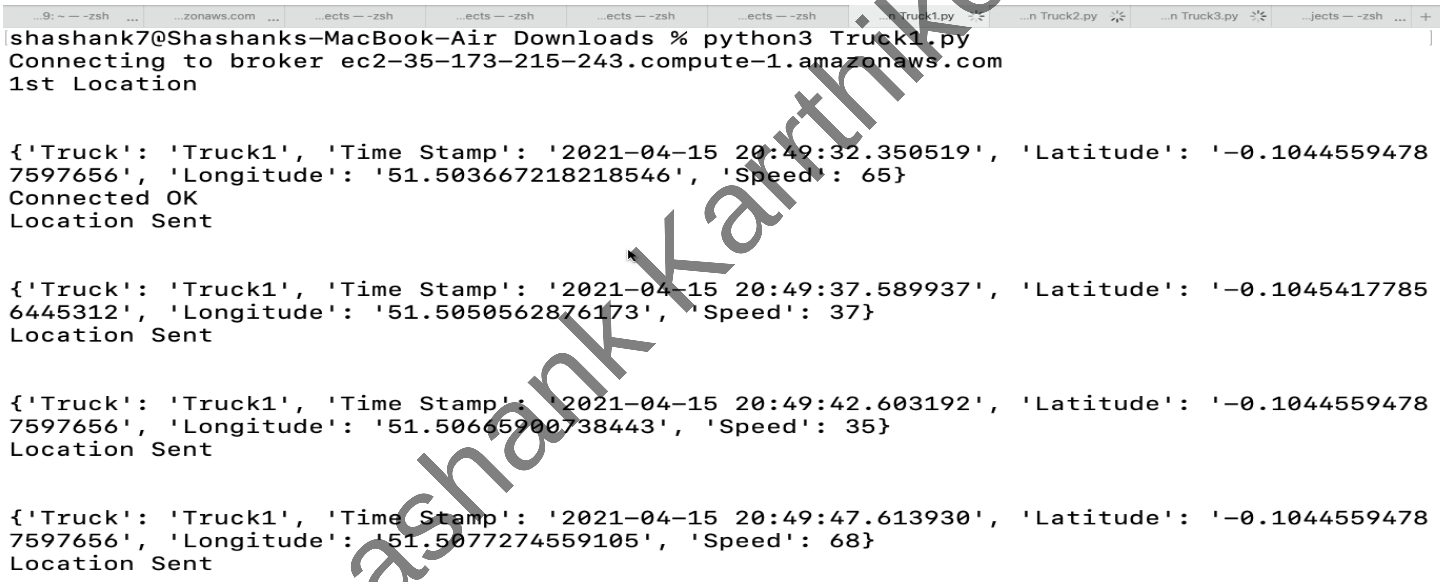
```
Truck2 :
  Time_Intreval_Seconds : 10
  Location :
    -   -0.12694358825683594 , 51.50767403407925
    -   -0.12303829193115234 , 51.50981085847289
    -   -0.11844635009765626 , 51.511600371790095
    -   -0.11492729187011717 , 51.51269541243519
    -   -0.11278152465820314 , 51.51349664501128
    -   -0.10934829711914062 , 51.51413762092547
    -   -0.10462760925292969 , 51.514217742280586
    -   -0.10432720184326172 , 51.51413762092547
    -   -0.10475635528564453 , 51.51614061252578
    -   -0.10514259338378906 , 51.51723554403216
    -   -0.11046409606933594 , 51.51806340159923
    -   -0.11587142944335936 , 51.51814351604911
    -   -0.1199054718017578 , 51.51766282723606
    -   -0.12050628662109375 , 51.51763612215324
    -   -0.11969089508056639 , 51.51632755391457
    -   -0.12106418609619139 , 51.51563319631723
    -   -0.12205123901367186 , 51.51467176063095
    -   -0.1242828369140625 , 51.51317615367198
    -   -0.12848854064941406 , 51.51125315828450
    -   -0.12831687927246094 , 51.51018479243817
    -   -0.12763023376464844 , 51.50949034120275
    -   -0.1272439956665039 , 51.508662327818094
    -   -0.12707233428955078 , 51.50780758853995
```

Figure 1.7: Config for Truck 2



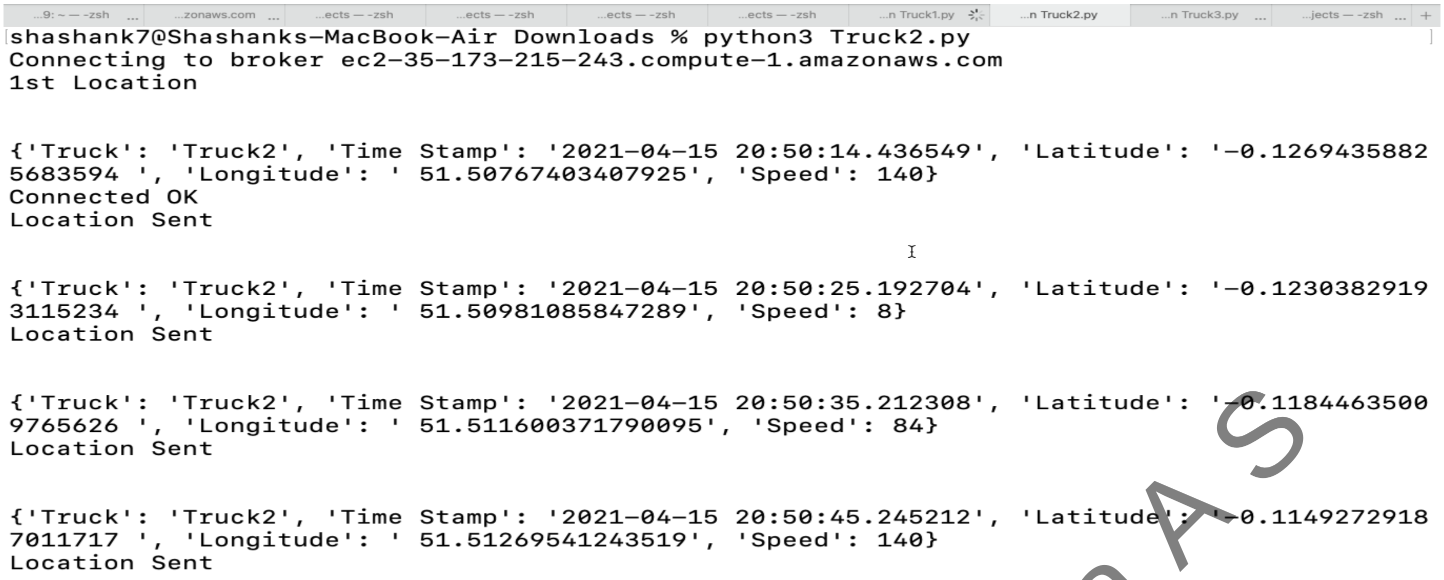
```
Truck3 :
  Time_Intreval_Seconds : 15
  Location :
    -  -0.12617111206054688 , 51.50099581189912
    -  -0.12402534484863281 , 51.50091566729007
    -  -0.12260913848876953 , 51.506285044481096
    -  -0.12076377868652344 , 51.50850206542486
    -  -0.11737346649169922 , 51.5102649207457
    -  -0.11346817016601562 , 51.51098606917176
    -  -0.10630130767822266 , 51.51101277815347
    -  -0.10342597961425781 , 51.511092905004745
    -  -0.10359764099121094 , 51.51296249152639
    -  -0.1036405563354492 , 51.5141376209254 7
    -  -0.10471343994140624 , 51.51419103517789
    -  -0.10647296905517578 , 51.51413762092547
    -  -0.10591506958007812 , 51.512882367963456
    -  -0.10677337646484375 , 51.51266870443993
    -  -0.10776042938232422 , 51.51427115643904
    -  -0.11432647705078125 , 51.51282895217651
    -  -0.11891841888427736 , 51.51133328471298
    -  -0.12196540832519531 , 51.5102649207457
    -  -0.1258707046508789 , 51.50836851299987
    -  -0.12767314910888672 , 51.507406923983446
    -  -0.12715816497802734 , 51.50583094254364
    -  -0.12629985809326172 , 51.50452203516731
    -  -0.12617111206054688 , 51.501262959578035
```

Figure 1.8: Config for Truck 3



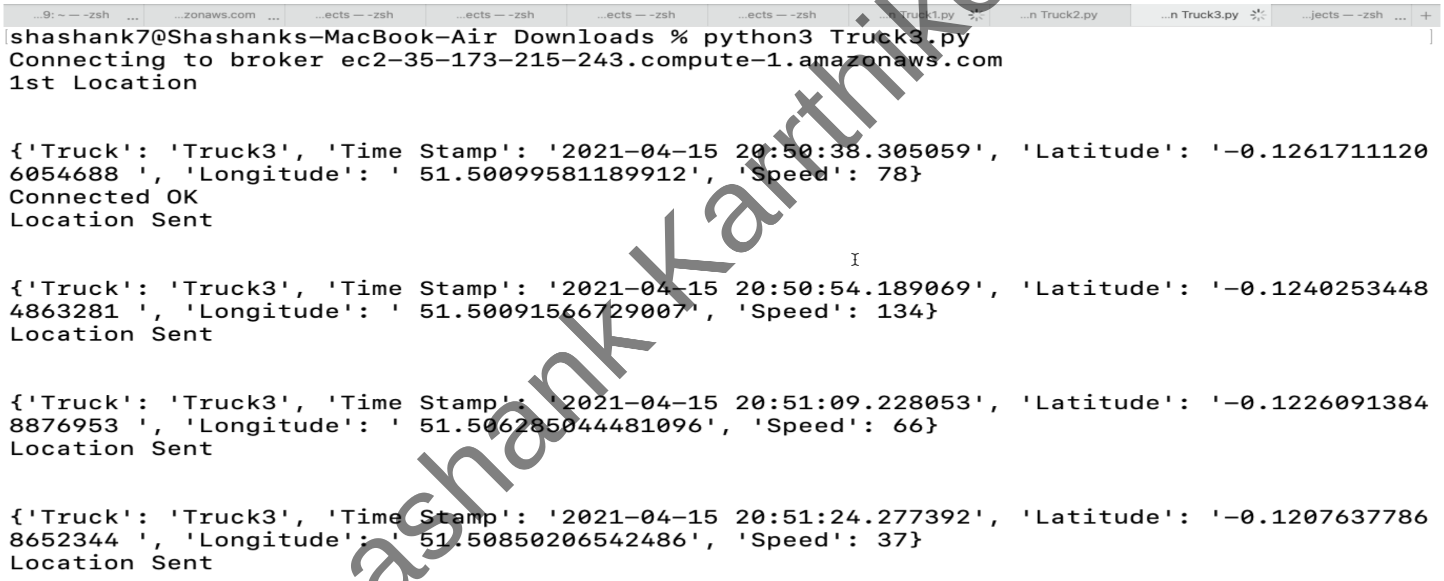
```
shashank7@Shashanks-MacBook-Air Downloads % python3 Truck1.py
Connecting to broker ec2-35-173-215-243.compute-1.amazonaws.com
1st Location


{'Truck': 'Truck1', 'Time Stamp': '2021-04-15 20:49:32.350519', 'Latitude': '-0.1044559478
7597656', 'Longitude': '51.503667218218546', 'Speed': 65}
Connected OK
Location Sent


{'Truck': 'Truck1', 'Time Stamp': '2021-04-15 20:49:37.589937', 'Latitude': '-0.1045417785
6445312', 'Longitude': '51.50505628761 73', 'Speed': 37}
Location Sent


{'Truck': 'Truck1', 'Time Stamp': '2021-04-15 20:49:42.603192', 'Latitude': '-0.1044559478
7597656', 'Longitude': '51.50665900738443', 'Speed': 35}
Location Sent


{'Truck': 'Truck1', 'Time Stamp': '2021-04-15 20:49:47.613930', 'Latitude': '-0.1044559478
7597656', 'Longitude': '51.5077274559105', 'Speed': 68}
Location Sent
```

Figure 1.9: Output for Truck 1



```
[shashank7@Shashanks-MacBook-Air Downloads % python3 Truck2.py
Connecting to broker ec2-35-173-215-243.compute-1.amazonaws.com
1st Location


{'Truck': 'Truck2', 'Time Stamp': '2021-04-15 20:50:14.436549', 'Latitude': '-0.1269435882
5683594 ', 'Longitude': ' 51.50767403407925', 'Speed': 140}
Connected OK
Location Sent


{'Truck': 'Truck2', 'Time Stamp': '2021-04-15 20:50:25.192704', 'Latitude': '-0.1230382919
3115234 ', 'Longitude': ' 51.50981085847289', 'Speed': 8}
Location Sent


{'Truck': 'Truck2', 'Time Stamp': '2021-04-15 20:50:35.212308', 'Latitude': '-0.1184463500
9765626 ', 'Longitude': ' 51.511600371790095', 'Speed': 84}
Location Sent


{'Truck': 'Truck2', 'Time Stamp': '2021-04-15 20:50:45.245212', 'Latitude': '-0.1149272918
7011717 ', 'Longitude': ' 51.51269541243519', 'Speed': 140}
Location Sent
```

Figure 1.10: Output for Truck 2



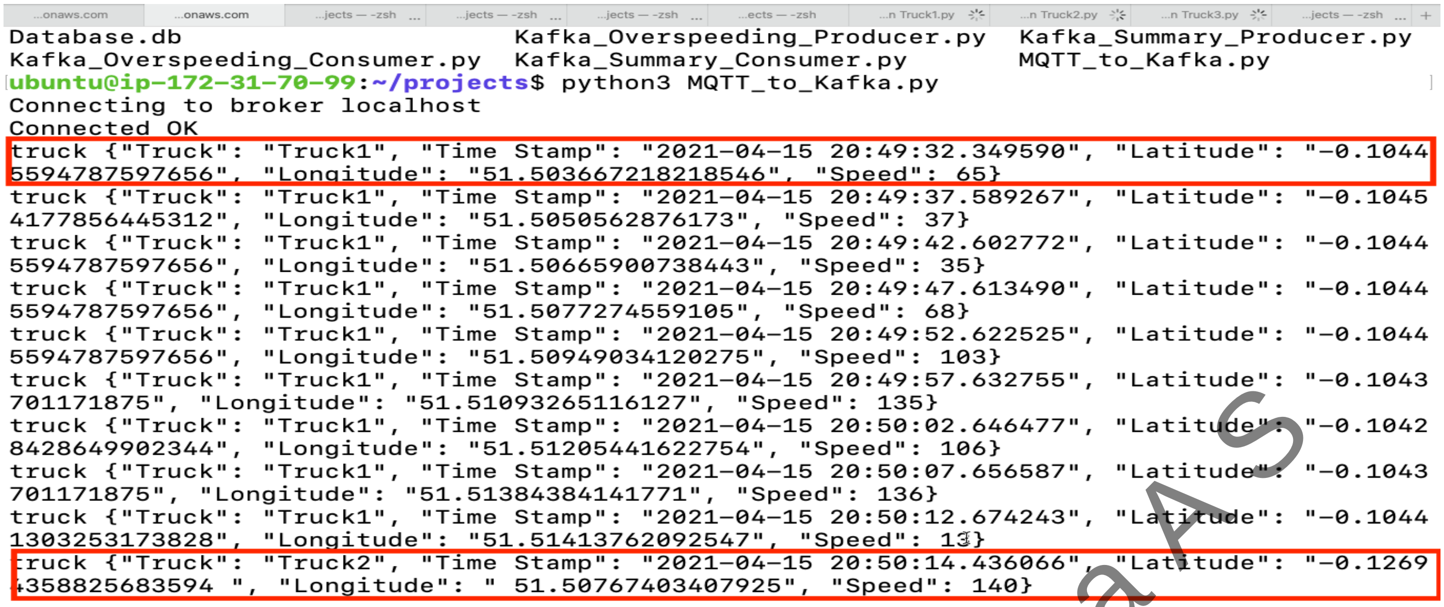
```
[shashank7@Shashanks-MacBook-Air Downloads % python3 Truck3.py
Connecting to broker ec2-35-173-215-243.compute-1.amazonaws.com
1st Location


{'Truck': 'Truck3', 'Time Stamp': '2021-04-15 20:50:38.305059', 'Latitude': '-0.1261711120
6054688 ', 'Longitude': ' 51.50099581189912', 'Speed': 78}
Connected OK
Location Sent


{'Truck': 'Truck3', 'Time Stamp': '2021-04-15 20:50:54.189069', 'Latitude': '-0.1240253448
4863281 ', 'Longitude': ' 51.50091566729007', 'Speed': 134}
Location Sent


{'Truck': 'Truck3', 'Time Stamp': '2021-04-15 20:51:09.228053', 'Latitude': '-0.1226091384
8876953 ', 'Longitude': ' 51.506285044481096', 'Speed': 66}
Location Sent


{'Truck': 'Truck3', 'Time Stamp': '2021-04-15 20:51:24.277392', 'Latitude': '-0.1207637786
8652344 ', 'Longitude': ' 51.50850206542486', 'Speed': 37}
Location Sent
```

Figure 1.11: Output for Truck 3

1.6 Simulater to MQTT, MQTT to Kafka

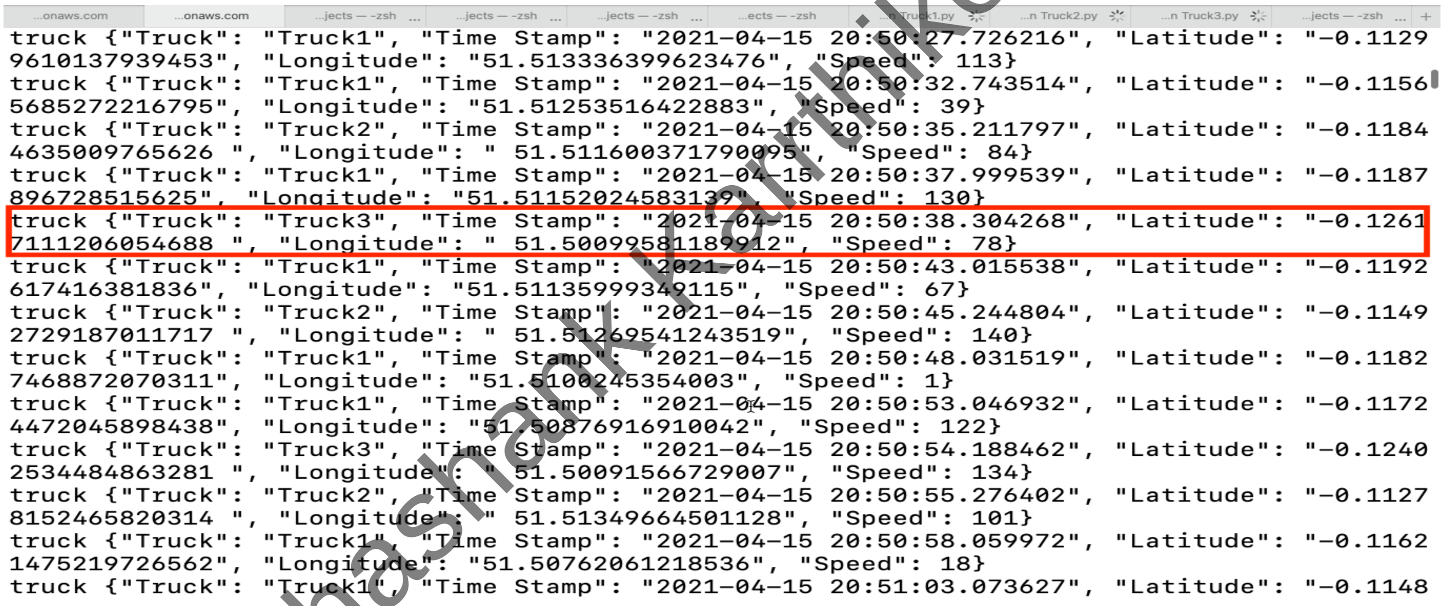
Python Script takes data from the trucks topic in the MQTT Broker and publishes the same under trucks topic in Kafka

The data from Trucks reaching the MQTT gateway is shown in the Figures 1.12 and 1.13



```
Database.db                         Kafka_Overspeeding_Producer.py   Kafka_Summary_Producer.py
Kafka_Overspeeding_Consumer.py      Kafka_Summary_Consumer.py        MQTT_to_Kafka.py
ubuntu@ip-172-31-70-99:~/projects$ python3 MQTT_to_Kafka.py
Connecting to broker localhost
Connected OK
truck {"Truck": "Truck1", "Time Stamp": "2021-04-15 20:49:32.349590", "Latitude": "-0.1044
5594787597656", "Longitude": "51.503667218218546", "Speed": 65}
truck {"Truck": "Truck1", "Time Stamp": "2021-04-15 20:49:37.589267", "Latitude": "-0.1045
4177856445312", "Longitude": "51.5050562876173", "Speed": 37}
truck {"Truck": "Truck1", "Time Stamp": "2021-04-15 20:49:42.602772", "Latitude": "-0.1044
5594787597656", "Longitude": "51.50665900738443", "Speed": 35}
truck {"Truck": "Truck1", "Time Stamp": "2021-04-15 20:49:47.613490", "Latitude": "-0.1044
5594787597656", "Longitude": "51.5077274559105", "Speed": 68}
truck {"Truck": "Truck1", "Time Stamp": "2021-04-15 20:49:52.622525", "Latitude": "-0.1044
5594787597656", "Longitude": "51.50949034120275", "Speed": 103}
truck {"Truck": "Truck1", "Time Stamp": "2021-04-15 20:49:57.632755", "Latitude": "-0.1043
701171875", "Longitude": "51.51093265116127", "Speed": 135}
truck {"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:02.646477", "Latitude": "-0.1042
8428649902344", "Longitude": "51.51205441622754", "Speed": 106}
truck {"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:07.656587", "Latitude": "-0.1043
701171875", "Longitude": "51.51384384141771", "Speed": 136}
truck {"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:12.674243", "Latitude": "-0.1044
1303253173828", "Longitude": "51.51413762092547", "Speed": 13}
truck {"Truck": "Truck2", "Time Stamp": "2021-04-15 20:50:14.436066", "Latitude": "-0.1269
4358825683594 ", "Longitude": " 51.50767403407925", "Speed": 140}
```

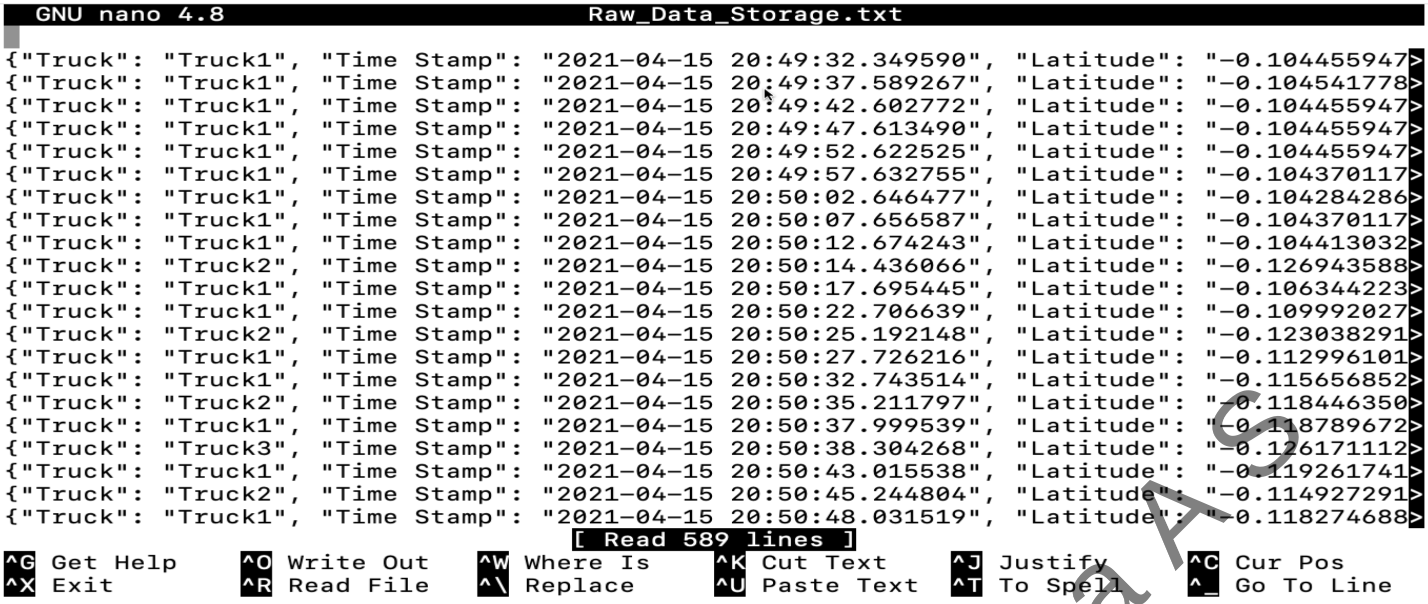
Figure 1.12: Data of Truck 1 and Truck 2



```
truck {"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:27.726216", "Latitude": "-0.1129
9610137939453", "Longitude": "51.513336399623476", "Speed": 113}
truck {"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:32.743514", "Latitude": "-0.1156
5685272216795", "Longitude": "51.51253516422883", "Speed": 39}
truck {"Truck": "Truck2", "Time Stamp": "2021-04-15 20:50:35.211797", "Latitude": "-0.1184
4635009765626 ", "Longitude": " 51.511600371790095", "Speed": 84}
truck {"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:37.999539", "Latitude": "-0.1187
896728515625", "Longitude": "51.51152024583139", "Speed": 130}
truck {"Truck": "Truck3", "Time Stamp": "2021-04-15 20:50:38.304268", "Latitude": "-0.1261
7111206054688 ", "Longitude": " 51.50099581189912", "Speed": 78}
truck {"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:43.015538", "Latitude": "-0.1192
617416381836", "Longitude": "51.51135999349115", "Speed": 67}
truck {"Truck": "Truck2", "Time Stamp": "2021-04-15 20:50:45.244804", "Latitude": "-0.1149
2729187011717 ", "Longitude": " 51.51269541243519", "Speed": 140}
truck {"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:48.031519", "Latitude": "-0.1182
7468872070311", "Longitude": "51.5100245354003", "Speed": 1}
truck {"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:53.046932", "Latitude": "-0.1172
4472045898438", "Longitude": "51.50876916910042", "Speed": 122}
truck {"Truck": "Truck3", "Time Stamp": "2021-04-15 20:50:54.188462", "Latitude": "-0.1240
2534484863281 ", "Longitude": " 51.50091566729007", "Speed": 134}
truck {"Truck": "Truck2", "Time Stamp": "2021-04-15 20:50:55.276402", "Latitude": "-0.1127
8152465820314 ", "Longitude": " 51.51349664501128", "Speed": 101}
truck {"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:58.059972", "Latitude": "-0.1162
1475219726562", "Longitude": "51.50762061218536", "Speed": 18}
truck {"Truck": "Truck1", "Time Stamp": "2021-04-15 20:51:03.073627", "Latitude": "-0.1148
```

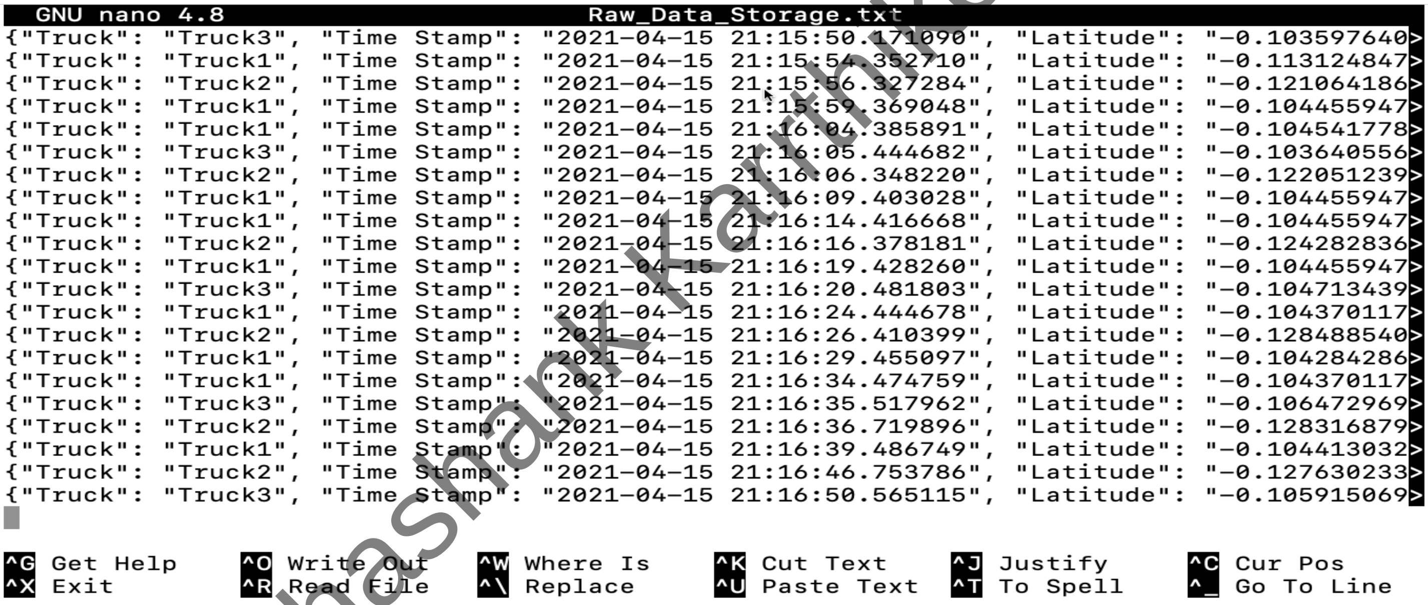
Figure 1.13: Data of Truck 3

The raw data stored in flat files can be seen in the Figures 1.14 and 1.15 . The data from the assets are stored as such without any parsing. If this file has to be processed then further parsing should be done.



```
  GNU nano 4.8                    Raw_Data_Storage.txt

{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:49:32.349590", "Latitude": "-0.104455947>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:49:37.589267", "Latitude": "-0.104541778>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:49:42.602772", "Latitude": "-0.104455947>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:49:47.613490", "Latitude": "-0.104455947>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:49:52.622525", "Latitude": "-0.104455947>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:49:57.632755", "Latitude": "-0.104370117>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:02.646477", "Latitude": "-0.104284286>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:07.656587", "Latitude": "-0.104370117>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:12.674243", "Latitude": "-0.104413032>
{"Truck": "Truck2", "Time Stamp": "2021-04-15 20:50:14.436066", "Latitude": "-0.126943588>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:17.695445", "Latitude": "-0.106344223>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:22.706639", "Latitude": "-0.109992027>
{"Truck": "Truck2", "Time Stamp": "2021-04-15 20:50:25.192148", "Latitude": "-0.123038291>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:27.726216", "Latitude": "-0.112996101>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:32.743514", "Latitude": "-0.115656852>
{"Truck": "Truck2", "Time Stamp": "2021-04-15 20:50:35.211797", "Latitude": "-0.118446350>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:37.999539", "Latitude": "-0.118789672>
{"Truck": "Truck3", "Time Stamp": "2021-04-15 20:50:38.304268", "Latitude": "-0.126171112>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:43.015538", "Latitude": "-0.119261741>
{"Truck": "Truck2", "Time Stamp": "2021-04-15 20:50:45.244804", "Latitude": "-0.114927291>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:48.031519", "Latitude": "-0.118274688>
                                   [ Read 589 lines ]
^G Get Help     ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos
^X Exit         ^R Read File    ^\ Replace      ^U Paste Text   ^T To Spell     ^_ Go To Line
```

Figure 1.14: Raw Data Starting



```
  GNU nano 4.8                    Raw_Data_Storage.txt
{"Truck": "Truck3", "Time Stamp": "2021-04-15 21:15:50.171090", "Latitude": "-0.103597640>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 21:15:54.352710", "Latitude": "-0.113124847>
{"Truck": "Truck2", "Time Stamp": "2021-04-15 21:15:56.327284", "Latitude": "-0.121064186>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 21:15:59.369048", "Latitude": "-0.104455947>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 21:16:04.385891", "Latitude": "-0.104541778>
{"Truck": "Truck3", "Time Stamp": "2021-04-15 21:16:05.444682", "Latitude": "-0.103640556>
{"Truck": "Truck2", "Time Stamp": "2021-04-15 21:16:06.348220", "Latitude": "-0.122051239>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 21:16:09.403028", "Latitude": "-0.104455947>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 21:16:14.416668", "Latitude": "-0.104455947>
{"Truck": "Truck2", "Time Stamp": "2021-04-15 21:16:16.378181", "Latitude": "-0.124282836>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 21:16:19.428260", "Latitude": "-0.104455947>
{"Truck": "Truck3", "Time Stamp": "2021-04-15 21:16:20.481803", "Latitude": "-0.104713439>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 21:16:24.444678", "Latitude": "-0.104370117>
{"Truck": "Truck2", "Time Stamp": "2021-04-15 21:16:26.410399", "Latitude": "-0.128488540>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 21:16:29.455097", "Latitude": "-0.104284286>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 21:16:34.474759", "Latitude": "-0.104370117>
{"Truck": "Truck3", "Time Stamp": "2021-04-15 21:16:35.517962", "Latitude": "-0.106472969>
{"Truck": "Truck2", "Time Stamp": "2021-04-15 21:16:36.719896", "Latitude": "-0.128316879>
{"Truck": "Truck1", "Time Stamp": "2021-04-15 21:16:39.486749", "Latitude": "-0.104413032>
{"Truck": "Truck2", "Time Stamp": "2021-04-15 21:16:46.753786", "Latitude": "-0.127630233>
{"Truck": "Truck3", "Time Stamp": "2021-04-15 21:16:50.565115", "Latitude": "-0.105915069>

^G Get Help     ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos
^X Exit         ^R Read File    ^\ Replace      ^U Paste Text   ^T To Spell     ^_ Go To Line
```

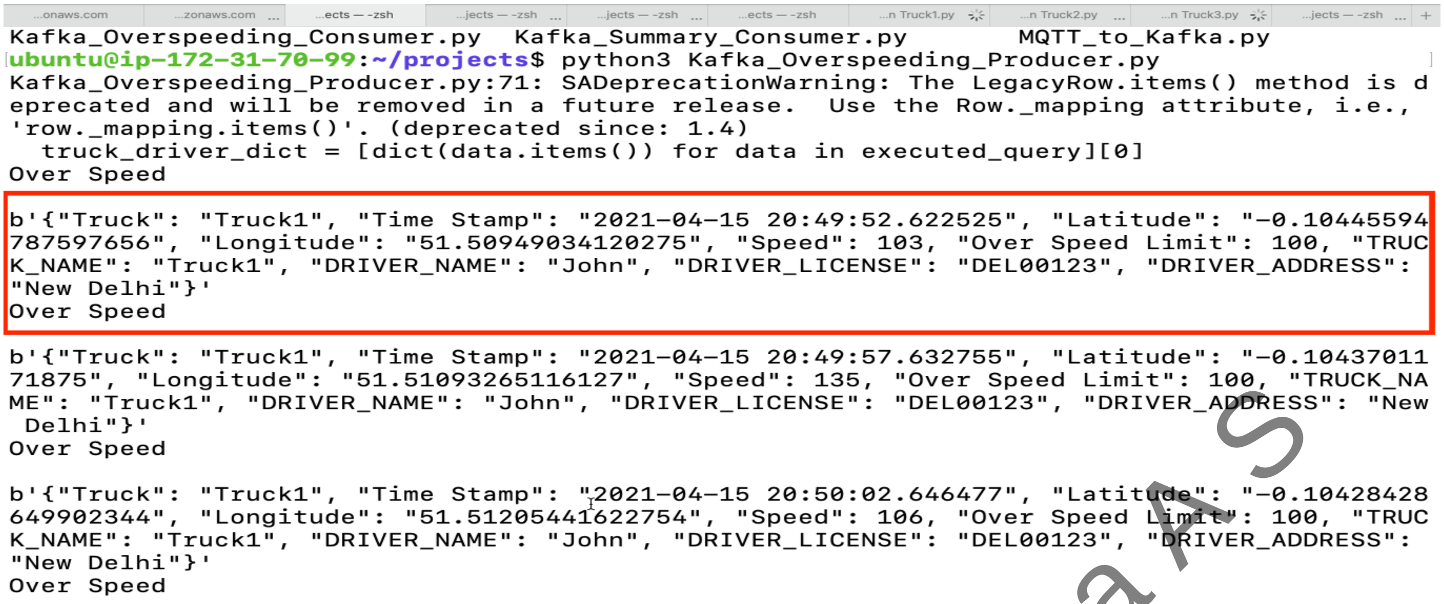
Figure 1.15: Raw Data Ending

1.7 Over Speeding Events

If Speed crossed 100 it is considered as Over speeding.
The producer and consumer for the Overspeeding events can be seen the figures 1.16, 1.18,1.20,1.17,1.19,1.21.

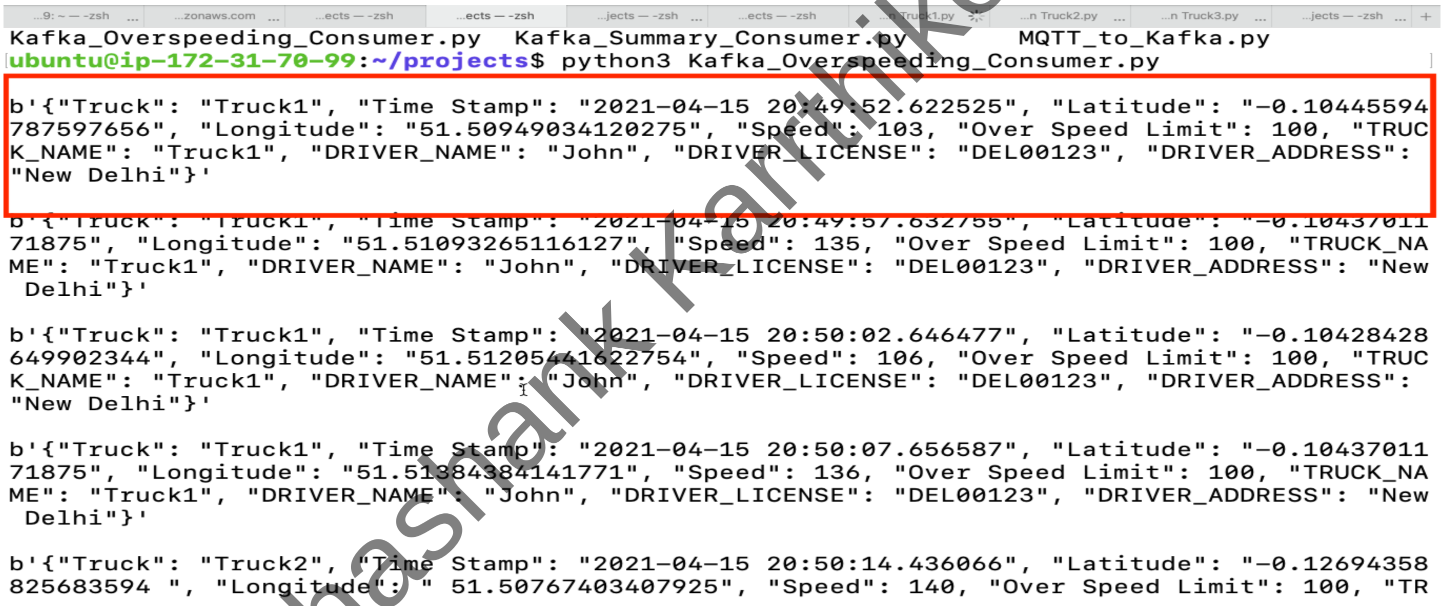
The figure 1.16 should be interpreted in the following way .

Truck 1 had a over speeding event at Latitude -0.104 and Longitude 51.509. When the over speeding event occured the speed was 103. The truck was driven by John whose license is DEL00123 and address is in New Delhi.



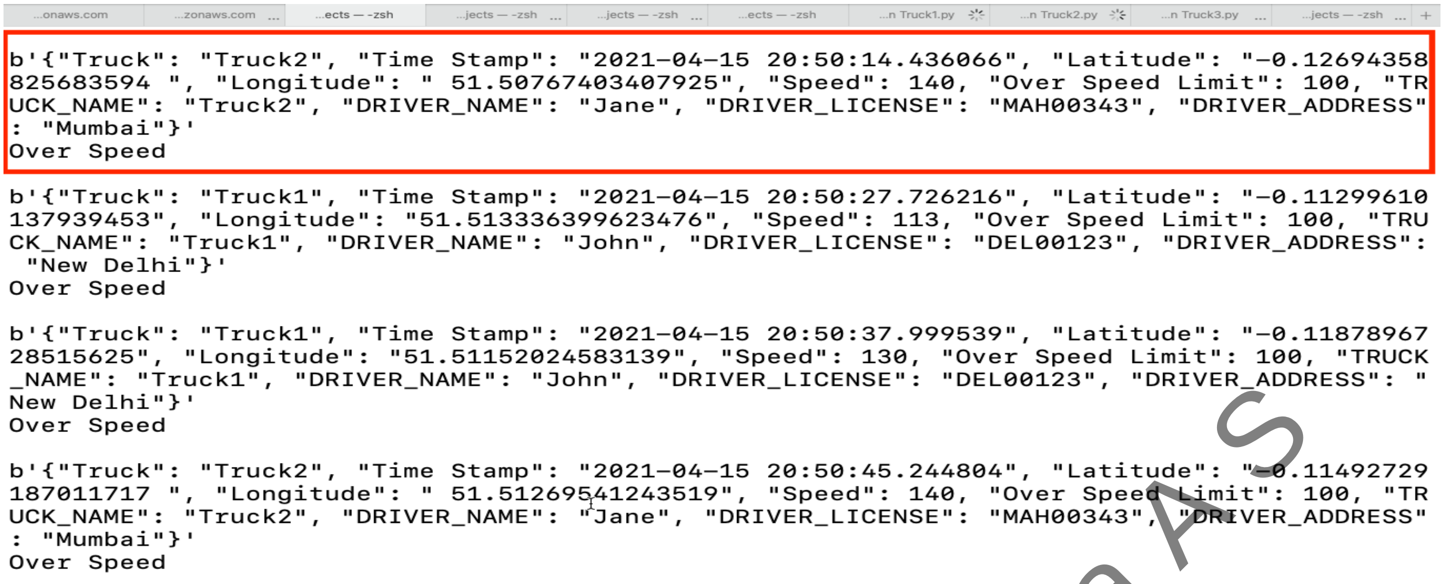
```
Kafka_Overspeeding_Consumer.py  Kafka_Summary_Consumer.py       MQTT_to_Kafka.py
ubuntu@ip-172-31-70-99:~/projects$ python3 Kafka_Overspeeding_Producer.py
Kafka_Overspeeding_Producer.py:71: SADeprecationWarning: The LegacyRow.items() method is d
eprecated and will be removed in a future release.  Use the Row._mapping attribute, i.e.,
'row._mapping.items()'. (deprecated since: 1.4)
  truck_driver_dict = [dict(data.items()) for data in executed_query][0]
Over Speed

b'{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:49:52.622525", "Latitude": "-0.10445594
787597656", "Longitude": "51.50949034120275", "Speed": 103, "Over Speed Limit": 100, "TRUC
K_NAME": "Truck1", "DRIVER_NAME": "John", "DRIVER_LICENSE": "DEL00123", "DRIVER_ADDRESS":
"New Delhi"}'
Over Speed

b'{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:49:57.632755", "Latitude": "-0.10437011
71875", "Longitude": "51.51093265116127", "Speed": 135, "Over Speed Limit": 100, "TRUCK_NA
ME": "Truck1", "DRIVER_NAME": "John", "DRIVER_LICENSE": "DEL00123", "DRIVER_ADDRESS": "New
 Delhi"}'
Over Speed

b'{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:02.646477", "Latitude": "-0.10428428
649902344", "Longitude": "51.51205441622754", "Speed": 106, "Over Speed Limit": 100, "TRUC
K_NAME": "Truck1", "DRIVER_NAME": "John", "DRIVER_LICENSE": "DEL00123", "DRIVER_ADDRESS":
"New Delhi"}'
Over Speed
```

Figure 1.16: Over Speeding Producer for Truck 1



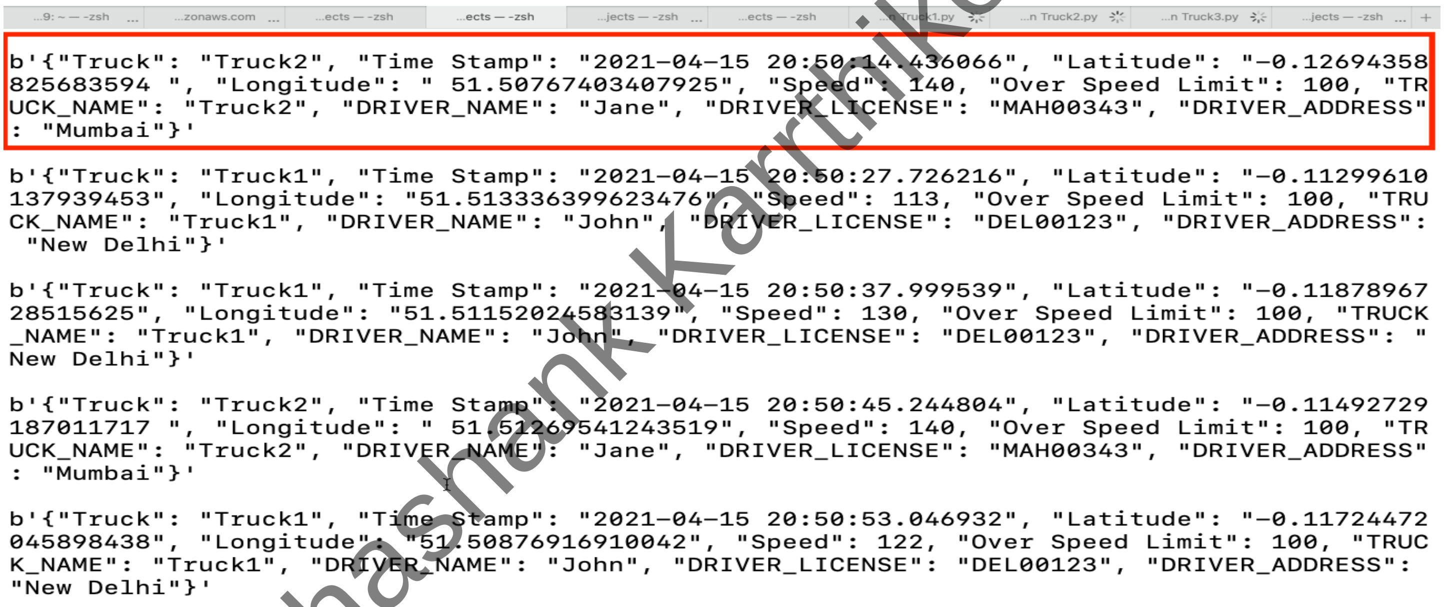
```
Kafka_Overspeeding_Consumer.py  Kafka_Summary_Consumer.py       MQTT_to_Kafka.py
ubuntu@ip-172-31-70-99:~/projects$ python3 Kafka_Overspeeding_Consumer.py

b'{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:49:52.622525", "Latitude": "-0.10445594
787597656", "Longitude": "51.50949034120275", "Speed": 103, "Over Speed Limit": 100, "TRUC
K_NAME": "Truck1", "DRIVER_NAME": "John", "DRIVER_LICENSE": "DEL00123", "DRIVER_ADDRESS":
"New Delhi"}'

b'{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:49:57.632755", "Latitude": "-0.10437011
71875", "Longitude": "51.51093265116127", "Speed": 135, "Over Speed Limit": 100, "TRUCK_NA
ME": "Truck1", "DRIVER_NAME": "John", "DRIVER_LICENSE": "DEL00123", "DRIVER_ADDRESS": "New
 Delhi"}'

b'{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:02.646477", "Latitude": "-0.10428428
649902344", "Longitude": "51.51205441622754", "Speed": 106, "Over Speed Limit": 100, "TRUC
K_NAME": "Truck1", "DRIVER_NAME": "John", "DRIVER_LICENSE": "DEL00123", "DRIVER_ADDRESS":
"New Delhi"}'

b'{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:07.656587", "Latitude": "-0.10437011
71875", "Longitude": "51.51384384141771", "Speed": 136, "Over Speed Limit": 100, "TRUCK_NA
ME": "Truck1", "DRIVER_NAME": "John", "DRIVER_LICENSE": "DEL00123", "DRIVER_ADDRESS": "New
 Delhi"}'

b'{"Truck": "Truck2", "Time Stamp": "2021-04-15 20:50:14.436066", "Latitude": "-0.12694358
825683594 ", "Longitude": " 51.50767403407925", "Speed": 140, "Over Speed Limit": 100, "TR
```

Figure 1.17: Over Speeding Consumer for Truck 1



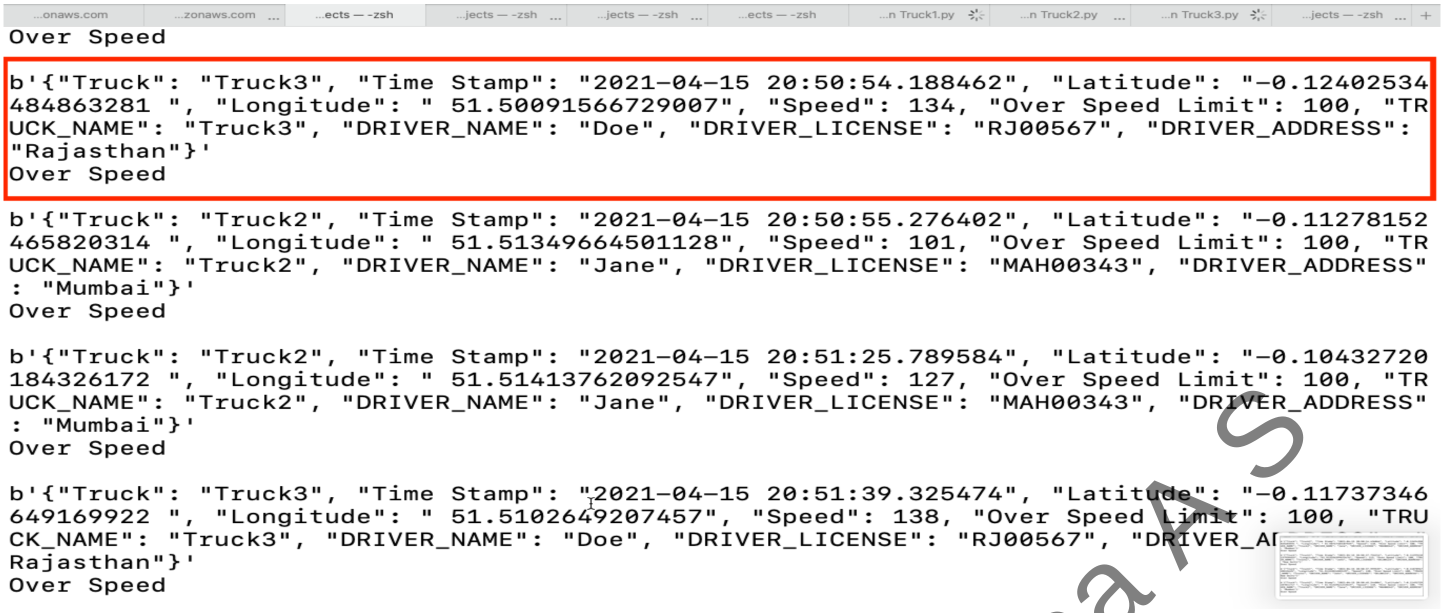
```
b'{"Truck": "Truck2", "Time Stamp": "2021-04-15 20:50:14.436066", "Latitude": "-0.12694358
825683594 ", "Longitude": " 51.50767403407925", "Speed": 140, "Over Speed Limit": 100, "TR
UCK_NAME": "Truck2", "DRIVER_NAME": "Jane", "DRIVER_LICENSE": "MAH00343", "DRIVER_ADDRESS"
: "Mumbai"}'
Over Speed

b'{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:27.726216", "Latitude": "-0.11299610
137939453", "Longitude": "51.513336399623476", "Speed": 113, "Over Speed Limit": 100, "TRU
CK_NAME": "Truck1", "DRIVER_NAME": "John", "DRIVER_LICENSE": "DEL00123", "DRIVER_ADDRESS":
 "New Delhi"}'
Over Speed

b'{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:37.999539", "Latitude": "-0.11878967
28515625", "Longitude": "51.51152024583139", "Speed": 130, "Over Speed Limit": 100, "TRUCK
_NAME": "Truck1", "DRIVER_NAME": "John", "DRIVER_LICENSE": "DEL00123", "DRIVER_ADDRESS": "
New Delhi"}'
Over Speed

b'{"Truck": "Truck2", "Time Stamp": "2021-04-15 20:50:45.244804", "Latitude": "-0.11492729
187011717 ", "Longitude": " 51.51269541243519", "Speed": 140, "Over Speed Limit": 100, "TR
UCK_NAME": "Truck2", "DRIVER_NAME": "Jane", "DRIVER_LICENSE": "MAH00343", "DRIVER_ADDRESS"
: "Mumbai"}'
Over Speed
```

Figure 1.18: Over Speeding Producer for Truck 2



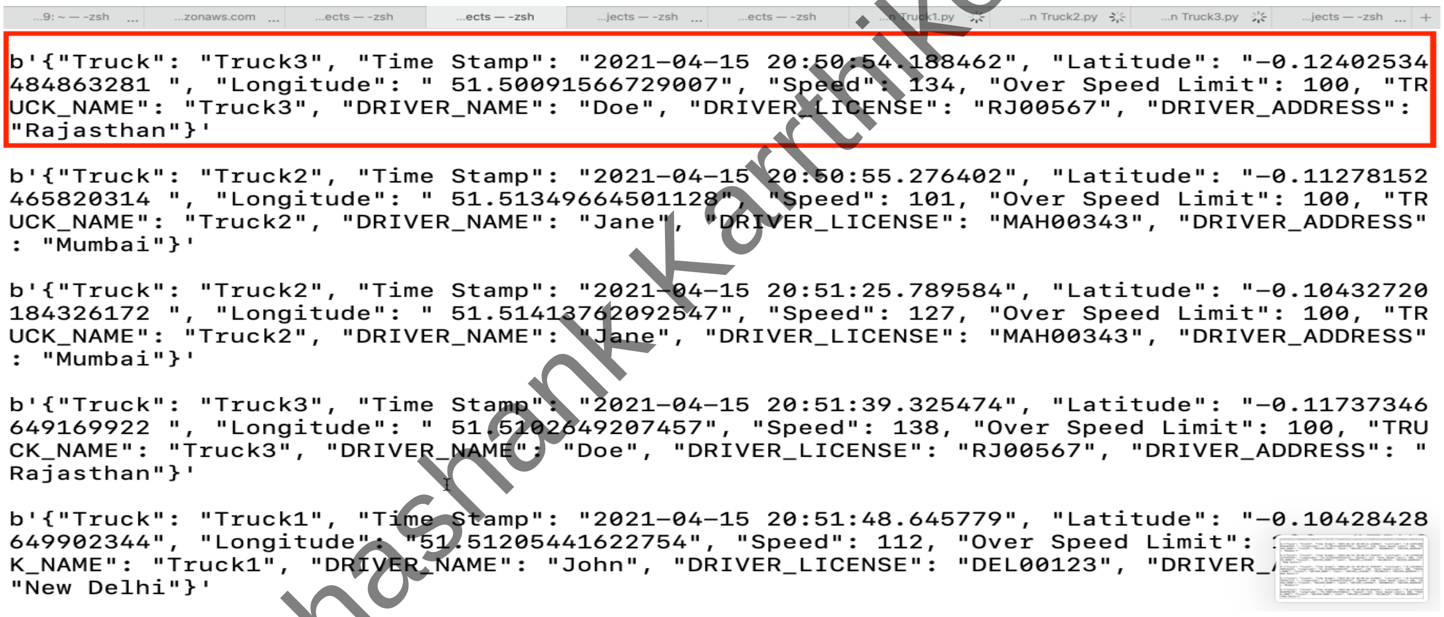
```
b'{"Truck": "Truck2", "Time Stamp": "2021-04-15 20:50:14.436066", "Latitude": "-0.12694358
825683594 ", "Longitude": " 51.50767403407925", "Speed": 140, "Over Speed Limit": 100, "TR
UCK_NAME": "Truck2", "DRIVER_NAME": "Jane", "DRIVER_LICENSE": "MAH00343", "DRIVER_ADDRESS"
: "Mumbai"}'

b'{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:27.726216", "Latitude": "-0.11299610
137939453", "Longitude": "51.513336399623476", "Speed": 113, "Over Speed Limit": 100, "TRU
CK_NAME": "Truck1", "DRIVER_NAME": "John", "DRIVER_LICENSE": "DEL00123", "DRIVER_ADDRESS":
 "New Delhi"}'

b'{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:37.999539", "Latitude": "-0.11878967
28515625", "Longitude": "51.51152024583139", "Speed": 130, "Over Speed Limit": 100, "TRUCK
_NAME": "Truck1", "DRIVER_NAME": "John", "DRIVER_LICENSE": "DEL00123", "DRIVER_ADDRESS": "
New Delhi"}'

b'{"Truck": "Truck2", "Time Stamp": "2021-04-15 20:50:45.244804", "Latitude": "-0.11492729
187011717 ", "Longitude": " 51.51269541243519", "Speed": 140, "Over Speed Limit": 100, "TR
UCK_NAME": "Truck2", "DRIVER_NAME": "Jane", "DRIVER_LICENSE": "MAH00343", "DRIVER_ADDRESS"
: "Mumbai"}'

b'{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:50:53.046932", "Latitude": "-0.11724472
045898438", "Longitude": "51.50876916910042", "Speed": 122, "Over Speed Limit": 100, "TRUC
K_NAME": "Truck1", "DRIVER_NAME": "John", "DRIVER_LICENSE": "DEL00123", "DRIVER_ADDRESS":
"New Delhi"}'
```

Figure 1.19: Over Speeding Consumer for Truck 2



```
Over Speed

b'{"Truck": "Truck3", "Time Stamp": "2021-04-15 20:50:54.188462", "Latitude": "-0.12402534
484863281 ", "Longitude": " 51.50091566729007", "Speed": 134, "Over Speed Limit": 100, "TR
UCK_NAME": "Truck3", "DRIVER_NAME": "Doe", "DRIVER_LICENSE": "RJ00567", "DRIVER_ADDRESS":
"Rajasthan"}'
Over Speed

b'{"Truck": "Truck2", "Time Stamp": "2021-04-15 20:50:55.276402", "Latitude": "-0.11278152
465820314 ", "Longitude": " 51.51349664501128", "Speed": 101, "Over Speed Limit": 100, "TR
UCK_NAME": "Truck2", "DRIVER_NAME": "Jane", "DRIVER_LICENSE": "MAH00343", "DRIVER_ADDRESS"
: "Mumbai"}'
Over Speed

b'{"Truck": "Truck2", "Time Stamp": "2021-04-15 20:51:25.789584", "Latitude": "-0.10432720
184326172 ", "Longitude": " 51.51413762092547", "Speed": 127, "Over Speed Limit": 100, "TR
UCK_NAME": "Truck2", "DRIVER_NAME": "Jane", "DRIVER_LICENSE": "MAH00343", "DRIVER_ADDRESS"
: "Mumbai"}'
Over Speed

b'{"Truck": "Truck3", "Time Stamp": "2021-04-15 20:51:39.325474", "Latitude": "-0.11737346
649169922 ", "Longitude": " 51.5102649207457", "Speed": 138, "Over Speed Limit": 100, "TRU
CK_NAME": "Truck3", "DRIVER_NAME": "Doe", "DRIVER_LICENSE": "RJ00567", "DRIVER_AD
Rajasthan"}'
Over Speed
```

Figure 1.20: Over Speeding Producer for Truck 3



```
b'{"Truck": "Truck3", "Time Stamp": "2021-04-15 20:50:54.188462", "Latitude": "-0.12402534
484863281 ", "Longitude": " 51.50091566729007", "Speed": 134, "Over Speed Limit": 100, "TR
UCK_NAME": "Truck3", "DRIVER_NAME": "Doe", "DRIVER_LICENSE": "RJ00567", "DRIVER_ADDRESS":
"Rajasthan"}'

b'{"Truck": "Truck2", "Time Stamp": "2021-04-15 20:50:55.276402", "Latitude": "-0.11278152
465820314 ", "Longitude": " 51.51349664501128", "Speed": 101, "Over Speed Limit": 100, "TR
UCK_NAME": "Truck2", "DRIVER_NAME": "Jane", "DRIVER_LICENSE": "MAH00343", "DRIVER_ADDRESS"
: "Mumbai"}'

b'{"Truck": "Truck2", "Time Stamp": "2021-04-15 20:51:25.789584", "Latitude": "-0.10432720
184326172 ", "Longitude": " 51.51413762092547", "Speed": 127, "Over Speed Limit": 100, "TR
UCK_NAME": "Truck2", "DRIVER_NAME": "Jane", "DRIVER_LICENSE": "MAH00343", "DRIVER_ADDRESS"
: "Mumbai"}'

b'{"Truck": "Truck3", "Time Stamp": "2021-04-15 20:51:39.325474", "Latitude": "-0.11737346
649169922 ", "Longitude": " 51.5102649207457", "Speed": 138, "Over Speed Limit": 100, "TRU
CK_NAME": "Truck3", "DRIVER_NAME": "Doe", "DRIVER_LICENSE": "RJ00567", "DRIVER_ADDRESS": "
Rajasthan"}'

b'{"Truck": "Truck1", "Time Stamp": "2021-04-15 20:51:48.645779", "Latitude": "-0.10428428
649902344", "Longitude": "51.51205441622754", "Speed": 112, "Over Speed Limit":
K_NAME": "Truck1", "DRIVER_NAME": "John", "DRIVER_LICENSE": "DEL00123", "DRIVER_
"New Delhi"}'
```

Figure 1.21: Over Speeding Consumer for Truck 3

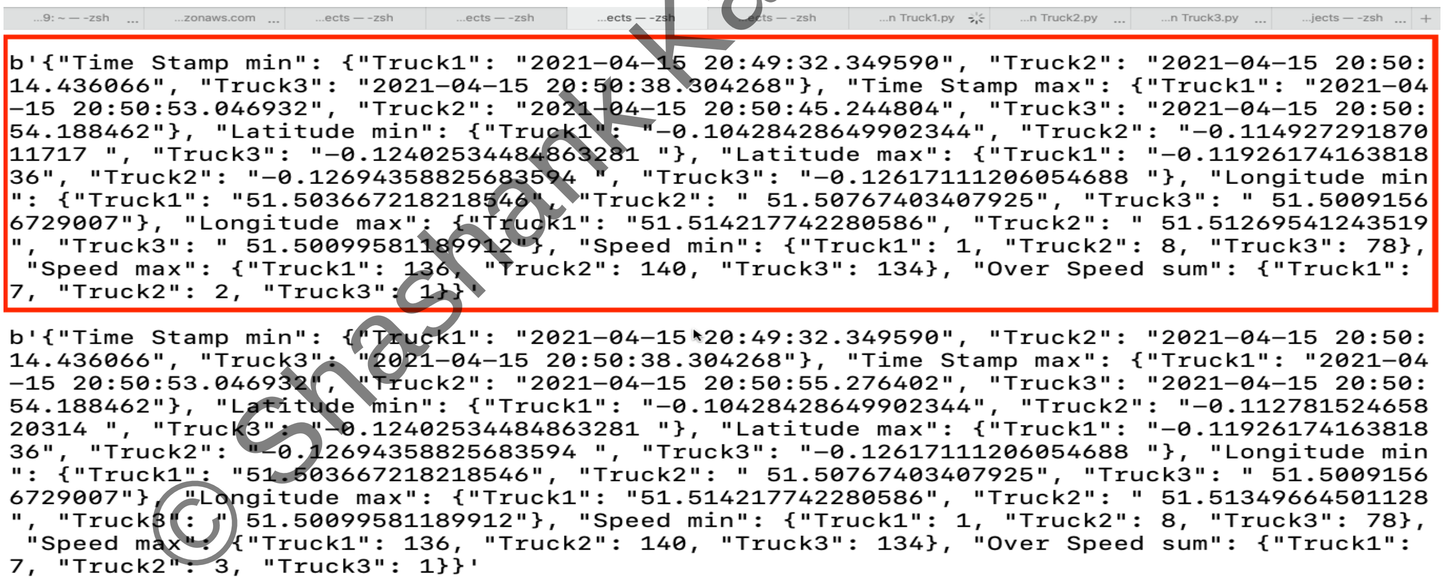
1.8 Data Summary

The Summary of the data is provided for every 2 Minutes. The summary can be changed by using the variable in the Script.

The Producer and Consumer in the Data Summary can be found in the Figures 1.22 and 1.23 respectively.

Fig. 1.22 should be interpreted in the following way

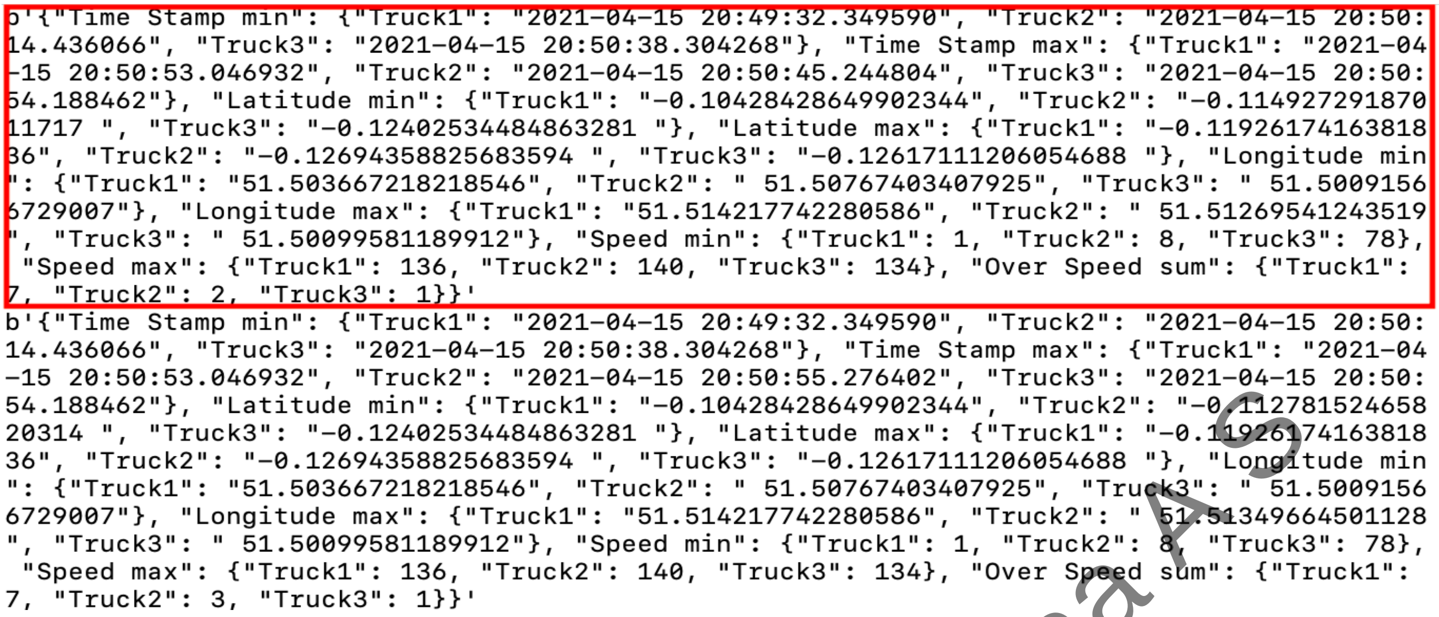
1. Truck 1
   (a) The truck has been sending data from 15th of April 2021 8:49 PM to 8:50 PM.
   (b) The Minimum Latitude of the Truck is -0.1014 and Maximum Latitude is -0.119
   (c) The Engine Speed ranges from 1 to 136
   (d) The truck had 7 Overspeed events
2. Truck 2
   (a) The truck has been sending data from 15th of April 2021 8:50 PM to 8:50 PM.
   (b) The Minimum Latitude of the Truck is -0.114 and Maximum Latitude is -0.1269
   (c) The Engine Speed ranges from 8 to 140
   (d) The truck had 2 Overspeed events
3. Truck 3
   (a) The truck has been sending data from 15th of April 2021 8:49 PM to 8:50 PM.
   (b) The Minimum Latitude of the Truck is -0.126 and Maximum Latitude is -0.126
   (c) The Engine Speed ranges from 78 to 134
   (d) The truck had 1 Overspeed events



```
b'{"Time Stamp min": {"Truck1": "2021-04-15 20:49:32.349590", "Truck2": "2021-04-15 20:50:
14.436066", "Truck3": "2021-04-15 20:50:38.304268"}, "Time Stamp max": {"Truck1": "2021-04
-15 20:50:53.046932", "Truck2": "2021-04-15 20:50:45.244804", "Truck3": "2021-04-15 20:50:
54.188462"}, "Latitude min": {"Truck1": "-0.10428428649902344", "Truck2": "-0.114927291870
11717 ", "Truck3": "-0.12402534484863281 "}, "Latitude max": {"Truck1": "-0.11926174163818
36", "Truck2": "-0.12694358825683594 ", "Truck3": "-0.12617111206054688 "}, "Longitude min
": {"Truck1": "51.50366721821854 6", "Truck2": " 51.50767403407925", "Truck3": " 51.5009156
6729007"}, "Longitude max": {"Truck1": "51.514217742280586", "Truck2": " 51.51269541243519
", "Truck3": " 51.50099581189912"}, "Speed min": {"Truck1": 1, "Truck2": 8, "Truck3": 78},
 "Speed max": {"Truck1": 136, "Truck2": 140, "Truck3": 134}, "Over Speed sum": {"Truck1":
7, "Truck2": 2, "Truck3": 1}}'
```

```
b'{"Time Stamp min": {"Truck1": "2021-04-15 20:49:32.349590", "Truck2": "2021-04-15 20:50:
14.436066", "Truck3": "2021-04-15 20:50:38.304268"}, "Time Stamp max": {"Truck1": "2021-04
-15 20:50:53.046932", "Truck2": "2021-04-15 20:50:55.276402", "Truck3": "2021-04-15 20:50:
54.188462"}, "Latitude min": {"Truck1": "-0.10428428649902344", "Truck2": "-0.112781524658
20314 ", "Truck3": "-0.12402534484863281 "}, "Latitude max": {"Truck1": "-0.1192617416381 8
36", "Truck2": "-0.12694358825683594 ", "Truck3": "-0.12617111206054688 "}, "Longitude min
": {"Truck1": "51.503667218218546", "Truck2": " 51.50767403407925", "Truck3": " 51.5009156
6729007"}, "Longitude max": {"Truck1": "51.514217742280586", "Truck2": " 51.51349664501128
", "Truck3": " 51.50099581189912"}, "Speed min": {"Truck1": 1, "Truck2": 8, "Truck3": 78},
 "Speed max": {"Truck1": 136, "Truck2": 140, "Truck3": 134}, "Over Speed sum": {"Truck1":
7, "Truck2": 3, "Truck3": 1}}'
```

Figure 1.22: Data Summary Producer



```
b'{"Time Stamp min": {"Truck1": "2021-04-15 20:49:32.349590", "Truck2": "2021-04-15 20:50:
14.436066", "Truck3": "2021-04-15 20:50:38.304268"}, "Time Stamp max": {"Truck1": "2021-04
-15 20:50:53.046932", "Truck2": "2021-04-15 20:50:45.244804", "Truck3": "2021-04-15 20:50:
54.188462"}, "Latitude min": {"Truck1": "-0.10428428649902344", "Truck2": "-0.114927291870
11717 ", "Truck3": "-0.12402534484863281 "}, "Latitude max": {"Truck1": "-0.11926174163818
36", "Truck2": "-0.12694358825683594 ", "Truck3": "-0.12617111206054688 "}, "Longitude min
": {"Truck1": "51.503667218218546", "Truck2": " 51.50767403407925", "Truck3": " 51.5009156
6729007"}, "Longitude max": {"Truck1": "51.514217742280586", "Truck2": " 51.51269541243519
", "Truck3": " 51.50099581189912"}, "Speed min": {"Truck1": 1, "Truck2": 8, "Truck3": 78},
 "Speed max": {"Truck1": 136, "Truck2": 140, "Truck3": 134}, "Over Speed sum": {"Truck1":
7, "Truck2": 2, "Truck3": 1}}'
b'{"Time Stamp min": {"Truck1": "2021-04-15 20:49:32.349590", "Truck2": "2021-04-15 20:50:
14.436066", "Truck3": "2021-04-15 20:50:38.304268"}, "Time Stamp max": {"Truck1": "2021-04
-15 20:50:53.046932", "Truck2": "2021-04-15 20:50:55.276402", "Truck3": "2021-04-15 20:50:
54.188462"}, "Latitude min": {"Truck1": "-0.10428428649902344", "Truck2": "-0.112781524658
20314 ", "Truck3": "-0.12402534484863281 "}, "Latitude max": {"Truck1": "-0.11926174163818
36", "Truck2": "-0.12694358825683594 ", "Truck3": "-0.12617111206054688 "}, "Longitude min
": {"Truck1": "51.503667218218546", "Truck2": " 51.50767403407925", "Truck3": " 51.5009156
6729007"}, "Longitude max": {"Truck1": "51.514217742280586", "Truck2": " 51.51349664501128
", "Truck3": " 51.50099581189912"}, "Speed min": {"Truck1": 1, "Truck2": 8, "Truck3": 78},
 "Speed max": {"Truck1": 136, "Truck2": 140, "Truck3": 134}, "Over Speed sum": {"Truck1":
7, "Truck2": 3, "Truck3": 1}}'
```

Figure 1.23: Data Summary Consumer

1.9 Conclusion

Thus a prototype of Intelligent Vehicle Monitoring System using Kafka has been implemented.

Appendix A

Simulator Code for Truck 1

```
# ————————————————————————————————————————————————————
# (C) Shashank Karrthikeyaa Annadanam Subbarathinam
# E–Mail : skas700@outlook.com
# GitHub : https://www.github.com/skas700
# ————————————————————————————————————————————————————

# IMporting required Libraries

#IMporting PyYaml to read the truck Configuration
import yaml

# Importing Sys,OS Packages for System functionalities
import sys,os

# Importing Date Time for Date & Time Operation
from datetime import datetime

# Importing MQTT Client
import paho.mqtt.client as mqtt

# Converting to JSON
import json

# Importing random Library to generate Random Integers
from random import randint

# ————————————————————————————————————————————————————

# Mentioning Broker Subscription and Publishing URL

# Address / IP of the location where the MQTT Broker is running
broker_address=”ec2−3−238−29−77.compute−1.amazonaws.com”
```

```
# URL to Subscribe
broker_subscribing_url=”test”

# URL to Publish
broker_publishing_url=”truck”

# Truck ID is the Node ID
node_id = ”Truck1”

# ————————————————————————————————————————————————————

# MQTT Functions

# Function to Log
def on_log(client,userdata,level,buf):
    pass

# Function to be executed when connected to the Broker
def on_connect(client,userdata,flags,rc):
    if rc==0:
        print(”Connected OK”)
        client.subscribe(broker_subscribing_url)
    else:
        print(”Bad connection Returned code=”,rc)

# Function to be executed on Disconnection
def on_disconnect(client,userdata,flags,rc=0):
    print(”DisConnected result code”+str(rc))

# Function to be executed when a mesage has arrived at the Subscribed Channel
def on_message(client,userdata,msg):
    topic=msg.topic
    m_decode=str(msg.payload.decode(”utf−8”))
    print(m_decode)

# Function publishing test details
def publishing_command_test():
    client.publish(broker_publishing_url,”Test”,qos=2)

# Function to Send Truck Location
def send_truck_location():

    global truck_last_sent_time
    global location_sent
    global location_list

    # Simulating Random Value of  Speed
```

```
speed_random_val = randint(0,140)

if truck_last_sent_time == None :

    print("1st Location")

    # Updating the Last Sent Time
    truck_last_sent_time = datetime.now()

    # Sending the First Location
    location_sent = 0

    # Getting the Latitude and Longitude from Location List
    latitude,longitude = location_list[location_sent].split(",")

    # Sending the required Information
    client.publish(broker_publishing_url,json.dumps(
            {"Truck":node_id,
            "Time Stamp":str(datetime.now()),
            "Latitude":latitude,
            "Longitude":longitude,
            "Speed":speed_random_val}))

    print("\n")
    print({"Truck":node_id,
            "Time Stamp":str(datetime.now()),
            "Latitude":latitude,
            "Longitude":longitude,
            "Speed":speed_random_val})

elif (datetime.now() - truck_last_sent_time).total_seconds() > time_intreval:

    print("Location Sent")

    # Updating the Last Sent Time
    truck_last_sent_time = datetime.now()

    if location_sent < len(location_list)-1:

        # Sending the Next Location
        location_sent = location_sent+1

    else:

        # Sending the Next Location
        location_sent = 0
        # Getting the Latitude and Longitude from Location List
        latitude,longitude = location_list[location_sent].split(”,”)

        # Sending the required Information
        client.publish(broker_publishing_url,json.dumps(
                {”Truck”:node_id,
                ”Time Stamp”:str(datetime.now()),
                ”Latitude”:latitude,
                ”Longitude”:longitude,
                ”Speed”:speed_random_val}))

        print(”\n”)
        print({”Truck”:node_id,
                ”Time Stamp”:str(datetime.now()),
                ”Latitude”:latitude,
                ”Longitude”:longitude,
                ”Speed”:speed_random_val})

        #print(truck_last_sent_time,location_sent)

    #return truck_last_sent_time,location_sent,location_list


# ————————————————————————————————————————————

# MAIN

# Loading Truck Configuration as a Python Dictionary
truckConfigDict = yaml.safe_load(open(os.getcwd()+”/truckConfig.yaml”))

# Printing Truck Configuration Dictionary
#print(truckConfigDict)

# Time Intreval for which Messages should be sent
time_intreval = truckConfigDict[node_id][’Time_Intreval_Seconds’]

# List of Locations
location_list = truckConfigDict[node_id][”Location”]

# ——————————

# Global Variable Keeping Track of Location Point sent
location_sent = None

# Global Variable Keeping Track of Last sent time
truck_last_sent_time = None
```

```
# ————————————

# Initializing MQTT Client
client=mqtt.Client(node_id,clean_session=True)

# Assigning Callback FUnction on Connecting
client.on_connect=on_connect

# Assigning Callback FUnction for Logging
client.on_log=on_log

# Assigning Callback FUnction on Disconnecting
client.on_disconnect=on_disconnect

# Assigning Callback FUnction to process received messages
client.on_message=on_message

# Printing Broker Address
print(”Connecting to broker”,broker_address)

# Connecting to the Broker
client.connect(broker_address)

# Looping and Sending Truck Information
while(1):
    client.loop()
    send_truck_location()

# Disconnecting from the Broker
client.disconnect()
```

Appendix B

Simulator Code for Truck 2

```
# ---------------------------------------------------------
# (C) Shashank Karrthikeyaa Annadanam Subbarathinam
# E-Mail : skas700@outlook.com
# GitHub : https://www.github.com/skas700
# ---------------------------------------------------------

# IMporting required Libraries

#IMporting PyYaml to read the truck Configuration
import yaml

# Importing Sys,OS Packages for System functionalities
import sys,os

# Importing Date Time for Date & Time Operation
from datetime import datetime

# Importing MQTT Client
import paho.mqtt.client as mqtt

# Converting to JSON
import json

# Importing random Library to generate Random Integers
from random import randint

# ---------------------------------------------------------

# Mentioning Broker Subscription and Publishing URL

# Address / IP of the location where the MQTT Broker is running
broker_address=”ec2-3-238-29-77.compute-1.amazonaws.com”

# URL to Subscribe
```

```
broker_subscribing_url=”test”

# URL to Publish
broker_publishing_url=”truck”

# Truck ID is the Node ID
node_id = ”Truck2”

# ------------------------------------------------------------

# MQTT Functions

# Function to Log
def on_log(client,userdata,level,buf):
    pass

# Function to be executed when connected to the Broker
def on_connect(client,userdata,flags,rc):
    if rc==0:
        print(”Connected OK”)
        client.subscribe(broker_subscribing_url)
    else:
        print(”Bad connection Returned code=”,rc)

# Function to be executed on Disconnection
def on_disconnect(client,userdata,flags,rc=0):
    print(”DisConnected result code”+str(rc))

# Function to be executed when a mesage has arrived at the Subscribed Channel
def on_message(client,userdata,msg):
    topic=msg.topic
    m_decode=str(msg.payload.decode(”utf−8”))
    print(m_decode)

# Function publishing test details
def publishing_command_test():
    client.publish(broker_publishing_url,”Test”,qos=2)

# Function to Send Truck Location
def send_truck_location():

    global truck_last_sent_time
    global location_sent
    global location_list

    # Simulating Random Value of  Speed
    speed_random_val = randint(0,140)
```

```
if truck_last_sent_time == None :

    print(”1st Location”)

    # Updating the Last Sent Time
    truck_last_sent_time = datetime.now()

    # Sending the First Location
    location_sent = 0

    # Getting the Latitude and Longitude from Location List
    latitude,longitude = location_list[location_sent].split(”,”)

    # Sending the required Information
    client.publish(broker_publishing_url,json.dumps(
            {”Truck”:node_id,
            ”Time Stamp”:str(datetime.now()),
            ”Latitude”:latitude,
            ”Longitude”:longitude,
            ”Speed”:speed_random_val}))

    print(”\n”)
    print({”Truck”:node_id,
            ”Time Stamp”:str(datetime.now()),
            ”Latitude”:latitude,
            ”Longitude”:longitude,
            ”Speed”:speed_random_val})

elif (datetime.now() - truck_last_sent_time).total_seconds() > time_intreval:

    print(”Location Sent”)

    # Updating the Last Sent Time
    truck_last_sent_time = datetime.now()

    if location_sent < len(location_list)-1:

        # Sending the Next Location
        location_sent = location_sent+1

    else:

        # Sending the Next Location
        location_sent = 0

    # Getting the Latitude and Longitude from Location List
```

```
        latitude,longitude = location_list[location_sent].split(”,”)

        # Sending the required Information
        client.publish(broker_publishing_url,json.dumps(
                {”Truck”:node_id,
                ”Time Stamp”:str(datetime.now()),
                ”Latitude”:latitude,
                ”Longitude”:longitude,
                ”Speed”:speed_random_val}))

        print(”\n”)
        print({”Truck”:node_id,
                ”Time Stamp”:str(datetime.now()),
                ”Latitude”:latitude,
                ”Longitude”:longitude,
                ”Speed”:speed_random_val})

        #print(truck_last_sent_time,location_sent)

    #return truck_last_sent_time,location_sent,location_list


# ——————————————————————————————————————————————————

# MAIN

# Loading Truck Configuration as a Python Dictionary
truckConfigDict = yaml.safe_load(open(os.getcwd()+”/truckConfig.yaml”))

# Printing Truck Configuration Dictionary
#print(truckConfigDict)

# Time Intreval for which Messages should be sent
time_intreval = truckConfigDict[node_id][’Time_Intreval_Seconds’]

# List of Locations
location_list = truckConfigDict[node_id][”Location”]

# ————————————

# Global Variable Keeping Track of Location Point sent
location_sent = None

# Global Variable Keeping Track of Last sent time
truck_last_sent_time = None

# ——————————————
# Initializing MQTT Client
client=mqtt.Client(node_id,clean_session=True)

# Assigning Callback FUnction on Connecting
client.on_connect=on_connect

# Assigning Callback FUnction for Logging
client.on_log=on_log

# Assigning Callback FUnction on Disconnecting
client.on_disconnect=on_disconnect

# Assigning Callback FUnction to process received messages
client.on_message=on_message

# Printing Broker Address
print("Connecting to broker",broker_address)

# Connecting to the Broker
client.connect(broker_address)

# Looping and Sending Truck Information
while(1):
    client.loop()
    send_truck_location()

# Disconnecting from the Broker
client.disconnect()
```

Appendix C

Simulator Code for Truck 3

```
# ------------------------------------------------------------
# (C) Shashank Karrthikeyaa Annadanam Subbarathinam
# E-Mail : skas700@outlook.com
# GitHub : https://www.github.com/skas700
# ------------------------------------------------------------

# IMporting required Libraries

#IMporting PyYaml to read the truck Configuration
import yaml

# Importing Sys,OS Packages for System functionalities
import sys,os

# Importing Date Time for Date & Time Operation
from datetime import datetime

# Importing MQTT Client
import paho.mqtt.client as mqtt

# Converting to JSON
import json

# Importing random Library to generate Random Integers
from random import randint

# ------------------------------------------------------------

# Mentioning Broker Subscription and Publishing URL

# Address / IP of the location where the MQTT Broker is running
broker_address=”ec2-3-238-29-77.compute-1.amazonaws.com”

# URL to Subscribe
```

```
broker_subscribing_url=”test”

# URL to Publish
broker_publishing_url=”truck”

# Truck ID is the Node ID
node_id = ”Truck3”

# ------------------------------------------------------------

# MQTT Functions

# Function to Log
def on_log(client,userdata,level,buf):
    pass

# Function to be executed when connected to the Broker
def on_connect(client,userdata,flags,rc):
    if rc==0:
        print(”Connected OK”)
        client.subscribe(broker_subscribing_url)
    else:
        print(”Bad connection Returned code=”,rc)

# Function to be executed on Disconnection
def on_disconnect(client,userdata,flags,rc=0):
    print(”DisConnected result code”+str(rc))

# Function to be executed when a mesage has arrived at the Subscribed Channel
def on_message(client,userdata,msg):
    topic=msg.topic
    m_decode=str(msg.payload.decode(”utf−8”))
    print(m_decode)

# Function publishing test details
def publishing_command_test():
    client.publish(broker_publishing_url,”Test”,qos=2)

# Function to Send Truck Location
def send_truck_location():

    global truck_last_sent_time
    global location_sent
    global location_list

    # Simulating Random Value of  Speed
    speed_random_val = randint(0,140)
```

```
if truck_last_sent_time == None :

    print(”1st Location”)

    # Updating the Last Sent Time
    truck_last_sent_time = datetime.now()

    # Sending the First Location
    location_sent = 0

    # Getting the Latitude and Longitude from Location List
    latitude,longitude = location_list[location_sent].split(”,”)

    # Sending the required Information
    client.publish(broker_publishing_url,json.dumps(
            {”Truck”:node_id,
            ”Time Stamp”:str(datetime.now()),
            ”Latitude”:latitude,
            ”Longitude”:longitude,
            ”Speed”:speed_random_val}))

    print(”\n”)
    print({”Truck”:node_id,
            ”Time Stamp”:str(datetime.now()),
            ”Latitude”:latitude,
            ”Longitude”:longitude,
            ”Speed”:speed_random_val})

elif (datetime.now() − truck_last_sent_time).total_seconds() > time_intreval:

    print(”Location Sent”)

    # Updating the Last Sent Time
    truck_last_sent_time = datetime.now()

    if location_sent < len(location_list)−1:

        # Sending the Next Location
        location_sent = location_sent+1

    else:

        # Sending the Next Location
        location_sent = 0

    # Getting the Latitude and Longitude from Location List
```

```
        latitude,longitude = location_list[location_sent].split(",")

        # Sending the required Information
        client.publish(broker_publishing_url,json.dumps(
                {"Truck":node_id,
                "Time Stamp":str(datetime.now()),
                "Latitude":latitude,
                "Longitude":longitude,
                "Speed":speed_random_val}))

        print("\n")
        print({"Truck":node_id,
                "Time Stamp":str(datetime.now()),
                "Latitude":latitude,
                "Longitude":longitude,
                "Speed":speed_random_val})

        #print(truck_last_sent_time,location_sent)

    #return truck_last_sent_time,location_sent,location_list


# ———————————————————————————————————————

# MAIN

# Loading Truck Configuration as a Python Dictionary
truckConfigDict = yaml.safe_load(open(os.getcwd()+"/truckConfig.yaml"))

# Printing Truck Configuration Dictionary
#print(truckConfigDict)

# Time Intreval for which Messages should be sent
time_intreval = truckConfigDict[node_id]['Time_Intreval_Seconds']

# List of Locations
location_list = truckConfigDict[node_id]["Location"]

# —————————

# Global Variable Keeping Track of Location Point sent
location_sent = None

# Global Variable Keeping Track of Last sent time
truck_last_sent_time = None

# ——————————
# Initializing MQTT Client
client=mqtt.Client(node_id,clean_session=True)

# Assigning Callback FUnction on Connecting
client.on_connect=on_connect

# Assigning Callback FUnction for Logging
client.on_log=on_log

# Assigning Callback FUnction on Disconnecting
client.on_disconnect=on_disconnect

# Assigning Callback FUnction to process received messages
client.on_message=on_message

# Printing Broker Address
print(”Connecting to broker”,broker_address)

# Connecting to the Broker
client.connect(broker_address)

# Looping and Sending Truck Information
while(1):
    client.loop()
    send_truck_location()

# Disconnecting from the Broker
client.disconnect()
```

Appendix D

Configuration for Trucks

```
Truck1 :
  Time_Intreval_Seconds : 5
  Location :
    - -0.10445594787597656,51.503667218218546
    - -0.10454177856445312,51.5050562876173
    - -0.10445594787597656,51.50665900738443
    - -0.10445594787597656,51.5077274559105
    - -0.10445594787597656,51.50949034120275
    - -0.1043701171875,51.51093265116127
    - -0.10428428649902344,51.51205441622754
    - -0.1043701171875,51.51384384141771
    - -0.10441303253173828,51.51413762092547
    - -0.10634422302246094,51.514217742280586
    - -0.10999202728271484,51.514030792232774
    - -0.11299610137939453,51.513336399623476
    - -0.11565685272216795,51.51253516422883
    - -0.1187896728515625,51.51152024583139
    - -0.1192617416381836,51.51135999349115
    - -0.11827468872070311,51.5100245354003
    - -0.11724472045898438,51.50876916910042
    - -0.11621475219726562,51.50762061218536
    - -0.11484146118164062,51.506285044481096
    - -0.11398315429687499,51.50537683608064
    - -0.11312484741210938,51.50502957514356

Truck2 :
  Time_Intreval_Seconds : 15
  Location :
    -  -0.12694358825683594 , 51.50767403407925
    -  -0.12303829193115234 , 51.50981085847289
    -  -0.11844635009765626 , 51.511600371790095
    -  -0.11492729187011717 , 51.51269541243519
    -  -0.11278152465820314 , 51.51349664501128
    -  -0.10934829711914062 , 51.51413762092547
    - −0.10462760925292969 , 51.514217742280586
    - −0.10432720184326172 , 51.51413762092547
    - −0.10475635528564453 , 51.51614061252578
    - −0.10514259338378906 , 51.51723554403216
    - −0.11046409606933594 , 51.51806340159923
    - −0.11587142944335936 , 51.51814351604911
    - −0.1199054718017578 , 51.51766282723606
    - −0.12050628662109375 , 51.51763612215324
    - −0.11969089508056639 , 51.51632755391457
    - −0.12106418609619139 , 51.51563319631723
    - −0.12205123901367186 , 51.51467176063095
    - −0.1242828369140625 , 51.51317615367198
    - −0.12848854064941406 , 51.5112531582845
    - −0.12831687927246094 , 51.51018479243817
    - −0.12763023376464844 , 51.50949034120275
    - −0.1272439956665039 , 51.508662327818094
    - −0.12707233428955078 , 51.50780758853995

Truck3 :
  Time_Intreval_Seconds : 10
  Location :
    - −0.12617111206054688 , 51.50099581189912
    - −0.12402534484863281 , 51.50091566729007
    - −0.12260913848876953 , 51.506285044481096
    - −0.12076377868652344 , 51.50850206542486
    - −0.11737346649169922 , 51.5102649207457
    - −0.11346817016601562 , 51.51098606917176
    - −0.10630130767822266 , 51.51101277815347
    - −0.10342597961425781 , 51.511092905004745
    - −0.10359764099121094 , 51.51296249152639
    - −0.1036405563354492 , 51.51413762092547
    - −0.10471343994140624 , 51.51419103517789
    - −0.10647296905517578 , 51.51413762092547
    - −0.10591506958007812 , 51.512882367963456
    - −0.10677337646484375 , 51.51266870443993
    - −0.10776042938232422 , 51.51427115643904
    - −0.11432647705078125 , 51.51282895217651
    - −0.11891841888427736 , 51.51133328471298
    - −0.12196540832519531 , 51.5102649207457
    - −0.1258707046508789 , 51.50836851299987
    - −0.12767314910888672 , 51.507406923983446
    - −0.12715816497802734 , 51.50583094254364
    - −0.12629985809326172 , 51.50452203516731
    - −0.12617111206054688 , 51.501262959578035
```

Appendix E

SQL Queries

```
— Query for Creating the Table
CREATE TABLE TRUCK_DETAILS(
TRUCK_NAME VARCHAR(50),
DRIVER_NAME VARCHAR(50),
DRIVER_LICENSE VARCHAR(50),
DRIVER_ADDRESS VARCHAR(100)
);

— Query for Inserting Data into the Tables
INSERT INTO TRUCK_DETAILS (TRUCK_NAME,DRIVER_NAME,DRIVER_LICENSE,DRIVER_ADDRESS)
VALUES
    ('Truck1','John','DEL00123','New Delhi'),
    ('Truck2','Jane','MAH00343','Mumbai'),
    ('Truck3','Doe','RJ00567','Rajasthan');

— Query for Selecting Data from the Tables
SELECT * FROM TRUCK_DETAILS;
```

Appendix F

MQTT to Kafka and Raw Data

```
# ————————————————————————————————————————————————————————
# (C) Shashank Karrthikeyaa Annadanam Subbarathinam
# E–Mail : skas700@outlook.com
# GitHub : https://www.github.com/skas700
# ————————————————————————————————————————————————————————

# Importing Required Packages

#IMporting PyYaml to read the truck Configuration
import yaml

# Importing Sys,OS Packages for System functionalities
import sys,os

# Importing MQTT Client
import paho.mqtt.client as mqtt

# Importing pyafka for connecting to kafka
from kafka import KafkaProducer

# ————————————————————————————————————————————————————————

# Configuration
# Server Address
server_address=”localhost”

# MQTT
mqtt_broker_subscribing_url=”truck”
mqtt_broker_publishing_url=”truck”
mqtt_node_id = ”Server”


# Kafka
kafka_publish_topic = ”truck”
```

```
# ---------------------------------------------------------------
# MQTT Functions

# Function to Log
def on_log(client,userdata,level,buf):
    pass

# Function to be executed when connected to the Broker
def on_connect(client,userdata,flags,rc):
    if rc==0:
        print("Connected OK")
        client.subscribe(mqtt_broker_subscribing_url)
    else:
        print("Bad connection Returned code=",rc)

# Function to be executed on Disconnection
def on_disconnect(client,userdata,flags,rc=0):
    print("DisConnected result code"+str(rc))

# Function to be executed when a mesage has arrived at the Subscribed Channel
def on_message(client,userdata,msg):
    topic=msg.topic
    m_decode=str(msg.payload.decode("utf-8"))
    print(topic,m_decode)

    # Inheriting Global Variables
    global kafka_producer
    global kafka_publish_topic

    # Forwarding message to Kafka Producer
    kafka_producer.send(kafka_publish_topic,msg.payload)

    # Initializinf File
    file = open("Raw Data Storage.txt","a")

    file.write(m_decode)

    file.close()

# ---------------------------------------------------------------

# Initializing Kafka Producer
kafka_producer = KafkaProducer(bootstrap_servers = [server_address+':9092'])

# Initializing MQTT Client
mqtt_client=mqtt.Client(mqtt_node_id,clean_session=True)
```

```
# Assigning Callback FUnction on Connecting
mqtt_client.on_connect=on_connect

# Assigning Callback FUnction for Logging
mqtt_client.on_log=on_log

# Assigning Callback FUnction on Disconnecting
mqtt_client.on_disconnect=on_disconnect

# Assigning Callback FUnction to process received messages
mqtt_client.on_message=on_message

# Printing Broker Address
print(”Connecting to broker”,server_address)

# Connecting to the Broker
mqtt_client.connect(server_address)

# Looping and Sending Truck Information
while(1):
    mqtt_client.loop()

# Disconnecting from the Broker
mqtt_client.disconnect()
```

Appendix G

Kafka Overspeeding Producer

```
# ————————————————————————————————————————————————————————
# (C) Shashank Karrthikeyaa Annadanam Subbarathinam
# E–Mail : skas700@outlook.com
# GitHub : https://www.github.com/skas700
# ————————————————————————————————————————————————————————

# Importing pyafka for connecting to kafka
from kafka import KafkaConsumer,KafkaProducer

# Importing JSON to convert JSON to Python Dictionary
import json

# Import SQLAlchemy
from sqlalchemy import create_engine

# Importing Pandas as Pd
import pandas as pd
# ————————————————————————————————————————————————————————

# Configuration
# Server Address
server_address=”localhost”

# Kafka
kafka_subscribe_topic = ”truck”
kafka_publish_topic = ”truck_overspeed”

# Overspeed limit
overspeed_limit = 100

# ————————————————————————————————————————————————————————
# Initializing Kafka Consumer
kafka_consumer = KafkaConsumer(kafka_subscribe_topic,bootstrap_servers = [server_a
# Initializing Kafka Producer
kafka_producer = KafkaProducer(bootstrap_servers = [server_address+':9092'])

# SQLAlchemy Engine
db_engine = create_engine(”sqlite:///Database.db”)

# Connection
db_Connection = db_engine.connect()

# Iterating through the messages
for message in kafka_consumer:

    #print(message.value)

    # Converting JSON to Python Dict
    message_dict = json.loads(message.value)

    # If Clause
    if message_dict[”Speed”] > overspeed_limit:

        # Getting Truck Driver Details
        truck_name = message_dict[”Truck”]

        # SQL Query
        query = '''
        SELECT
        *
        FROM
        TRUCK_DETAILS
        WHERE TRUCK_NAME = '{truck_name}'
         '''.format(truck_name = truck_name)

        # Executing the Query
        executed_query = db_Connection.execute(query)

        # Truck Driver Dictionary
        truck_driver_dict = [dict(data.items()) for data in executed_query][0]

        # print(truck_driver_dict)

        # Adding Over Speed Limit Value
        message_dict[”Over Speed Limit”] = overspeed_limit

        # Merging both Dictionaries
        for key in truck_driver_dict.keys():
            message_dict[key] = truck_driver_dict[key]
```

```
print(”Over Speed”)
print(””)

# Overspeed Dictionary
overspeed_json = json.dumps(message_dict)

# Converting to Bytes
overspeed_json = bytes(overspeed_json,’utf−8’)
print(overspeed_json)

# Forwarding message to Kafka Producer
kafka_producer.send(kafka_publish_topic,overspeed_json)
```

Appendix H

Kafka Overspeeding Consumer

```
# ------------------------------------------------------------------
# (C) Shashank Karrthikeyaa Annadanam Subbarathinam
# E–Mail : skas700@outlook.com
# GitHub : https://www.github.com/skas700
# ------------------------------------------------------------------

# Importing pyafka for connecting to kafka
from kafka import KafkaConsumer,KafkaProducer

# Importing JSON to convert JSON to Python Dictionary
import json

# ------------------------------------------------------------------

# Configuration
# Server Address
server_address=”localhost”

# Kafka
kafka_subscribe_topic = ”truck_overspeed”

# ------------------------------------------------------------------

# Initializing Kafka Consumer
kafka_consumer = KafkaConsumer(kafka_subscribe_topic,bootstrap_servers = [server_a

# Iterating through the messages
for message in kafka_consumer:

    print(””)
    print(message.value)
```

Appendix I

Kafka Summary Producer

```
# ————————————————————————————————————————————————————————————
# (C) Shashank Karrthikeyaa Annadanam Subbarathinam
# E–Mail : skas700@outlook.com
# GitHub : https://www.github.com/skas700
# ————————————————————————————————————————————————————————————

# Importing pyafka for connecting to kafka
from kafka import KafkaConsumer,KafkaProducer

# Importing JSON to convert JSON to Python Dictionary
import json

# Importing Pandas
import pandas as pd

# Importing Datetime
from datetime import datetime

# ————————————————————————————————————————————————————————————
# Configuration
# Server Address
server_address=”localhost”

# Kafka
kafka_subscribe_topic = ”truck”
kafka_publish_topic = ”truck_summary”

# Send Summary Time Seconds
send_summary_time_intreval_seconds = 120

# Overspeed limit
overspeed_limit = 100

# ————————————————————————————————————————————————————————————
# Initializing Dataframe for daily events
data_df = pd.DataFrame()

# Initializing Time Metric
last_updated_time = datetime(1997,1,1)

# ------------------------------------------------------------
# Initializing Kafka Consumer
kafka_consumer = KafkaConsumer(kafka_subscribe_topic ,bootstrap_servers = [server_a

# Initializing Kafka Producer
kafka_producer = KafkaProducer(bootstrap_servers = [server_address+’:9092’])

# Iterating through the messages
for message in kafka_consumer:

    # Converting JSON to Python Dict
    message_dict = json.loads(message.value)

    # Converting Dictionary to Pandas Data Frame
    message_df = pd.DataFrame(message_dict ,index = [0])

    # Appending to the DataFrame
    data_df = data_df.append(other = message_df ,ignore_index = True).reset_index(d

    if (datetime.now()-last_updated_time).total_seconds()>send_summary_time_intrev

        # Engine Overspeed element
        data_df[”Over Speed”] = data_df[”Speed”].apply(lambda x: 1 if x >= overspe


        # Grouping Information based on the Truck
        grouped_df = data_df.groupby(by=[”Truck”]).agg({”Time Stamp”:[”min”,”max”]
                                            ”Latitude”:[”min”,”max”],
                                            ”Longitude”:[”min”,”max”],
                                            ”Speed”:[”min”,”max”],
                                            ”Over Speed”:[”sum”]})


        #print(grouped_df)

        # Converting to Dictionary
        data_dict = grouped_df.to_dict()

        # Initializing New Dictionary
        data_dict_cleaned = {}
```

```
# Iterating through the keys
for key in data_dict.keys():
    new_key = ” ”.join(key)
    data_dict_cleaned[new_key] = data_dict[key]

# Overspeed Dictionary
summary_json = json.dumps(data_dict_cleaned)

# Converting to Bytes
summary_json = bytes(summary_json,’utf−8’)

print(””)
print(summary_json)

# Forwarding message to Kafka Producer
kafka_producer.send(kafka_publish_topic,summary_json)
```

Appendix J

Kafka Summary Consumer

```
# ————————————————————————————————————————————————
# (C) Shashank Karrthikeyaa Annadanam Subbarathinam
# E–Mail : skas700@outlook.com
# GitHub : https://www.github.com/skas700
# ————————————————————————————————————————————————

# Importing pyafka for connecting to kafka
from kafka import KafkaConsumer,KafkaProducer

# Importing JSON to convert JSON to Python Dictionary
import json

# ————————————————————————————————————————————————

# Configuration
# Server Address
server_address=”localhost”

# Kafka
kafka_subscribe_topic = ”truck_summary”

# ————————————————————————————————————————————————

# Initializing Kafka Consumer
kafka_consumer = KafkaConsumer(kafka_subscribe_topic,bootstrap_servers = [server_a

# Iterating through the messages
for message in kafka_consumer:

    print(message.value)
```