“Instructions: Make a working Big Data Streaming Pipeline and Integration Platform”

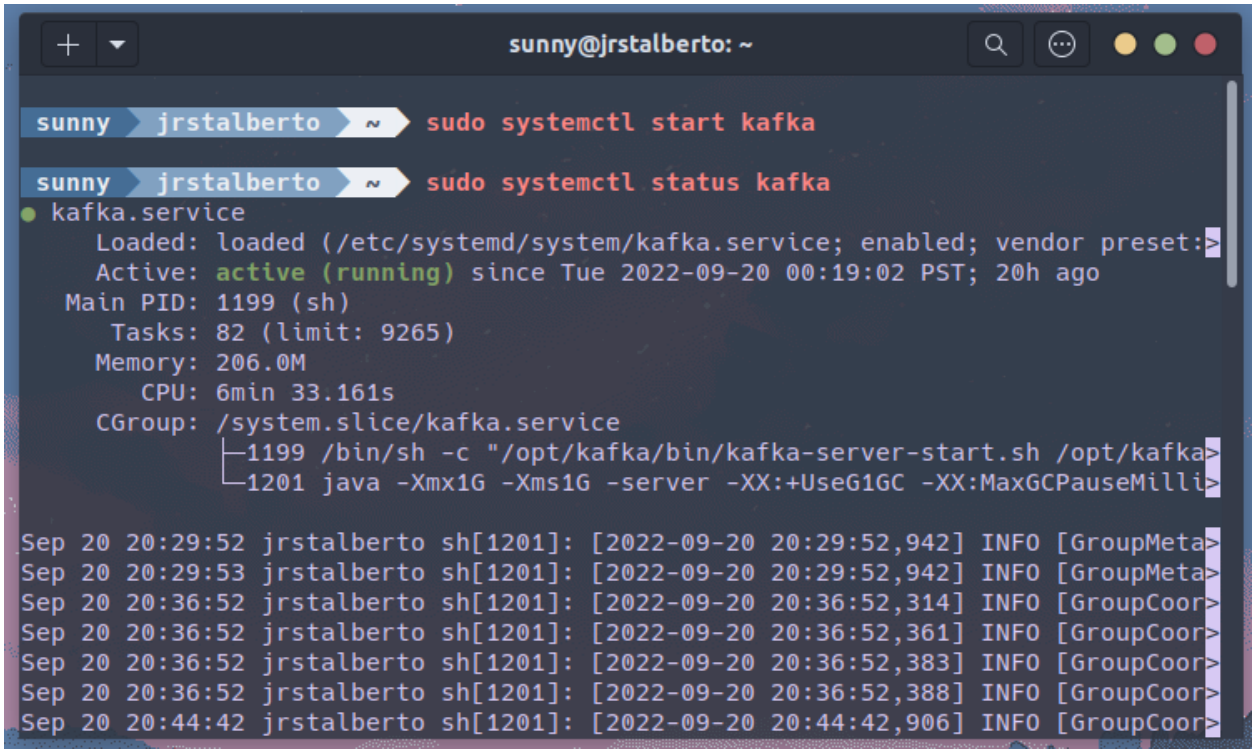
Initialize Kafka

Start kafka from the terminal using

```
sudo systemctl start kafka
```

And check if kafka started successfully

```
sudo systemctl status kafka
```

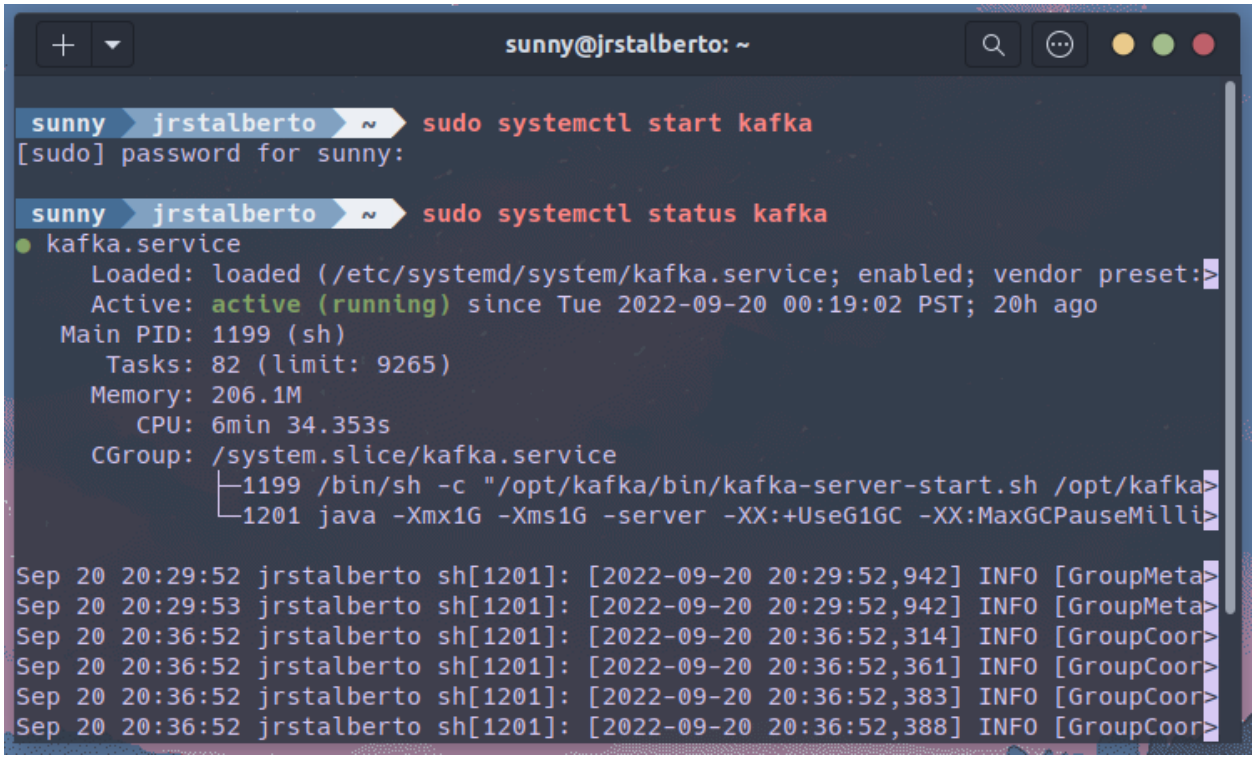


Open a new terminal instance then start kafka from the terminal

```
sudo systemctl start zookeeper
```

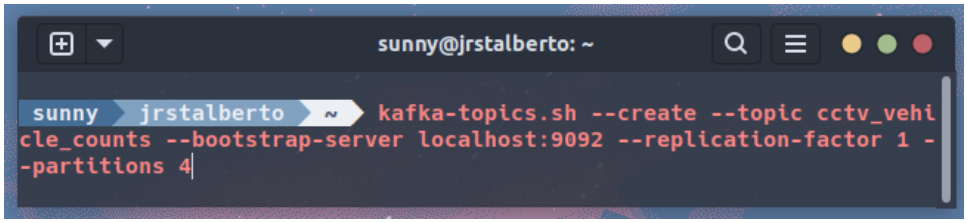
And check if zookeeper started successfully

```
sudo systemctl status zookeeper
```



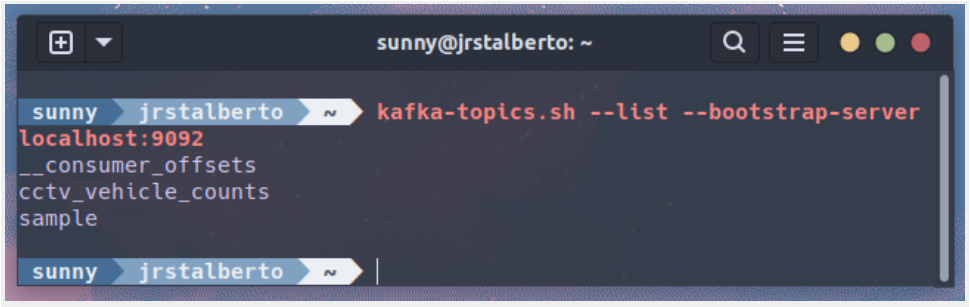
Create the topic that will be used for this project

```
kafka-topics.sh --create --topic <topic name> --bootstrap-server
localhost:9092 --replication-factor 1 --partitions 4
```



Verify if the topic was created successfully by listing the topics

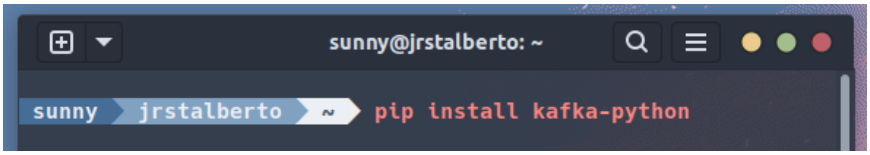
```
kafka-topics.sh --list --bootstrap-server localhost:9092
```



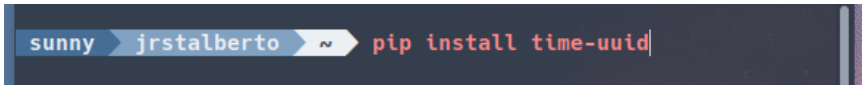
Prepare the consumer and producer programs

Install the required packages for the downloaded scripts

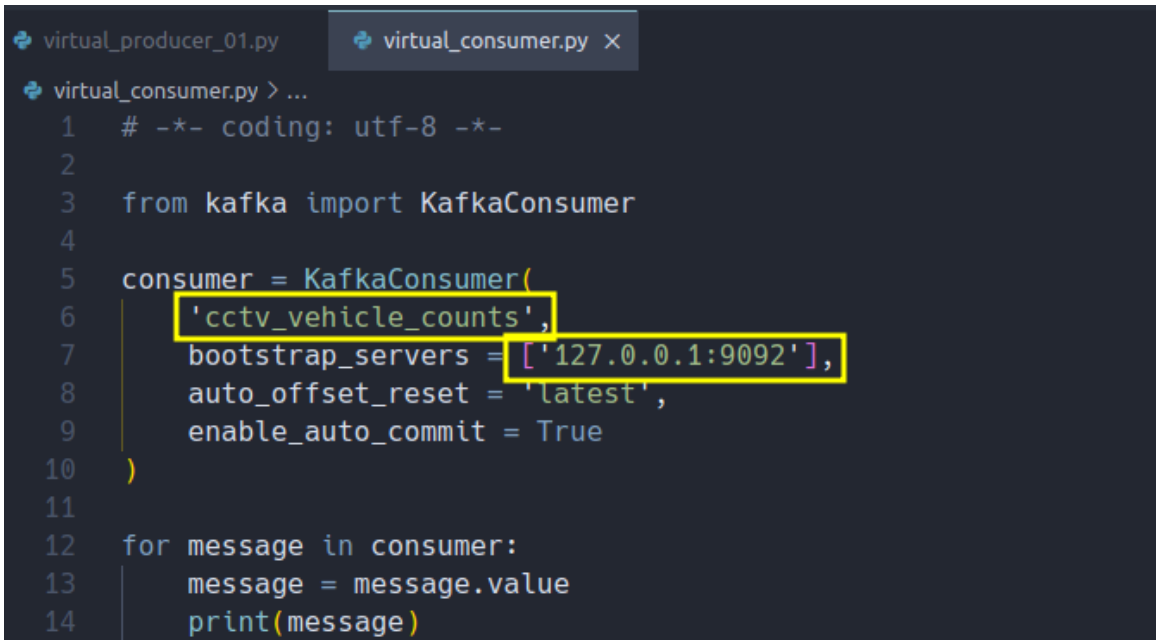
```
pip install kafka-python
```



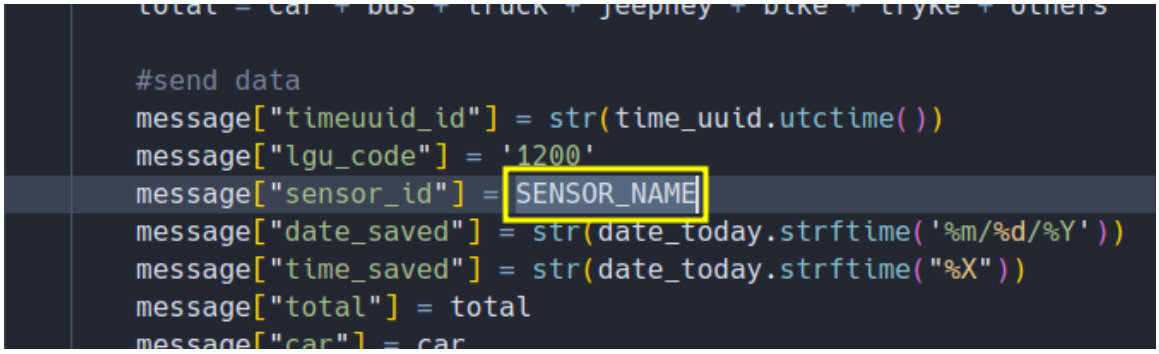

```
pip install time-uuid
```



From the given consumer file, ensure that the topic name and bootstrap server is correct.



From the given producer file, replace the value of sensor_id with the variable. SENSOR_NAME



Create the variable SENSOR_NAME and supply the appropriate value.

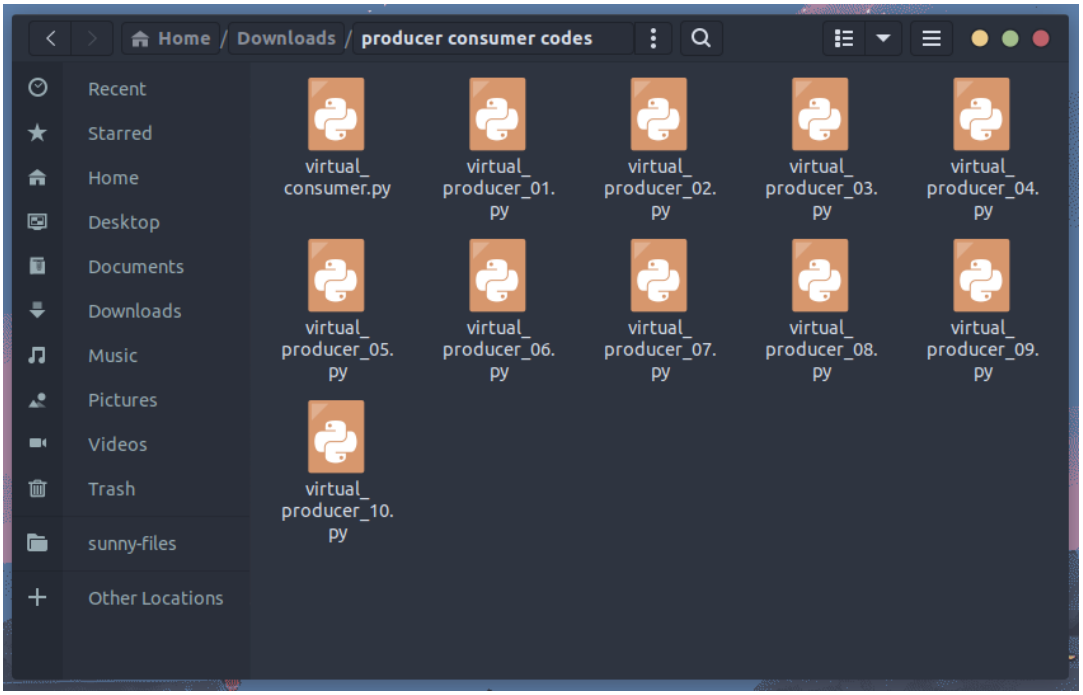
Verify that the topic name and server are correct.

```
import random

import time_uuid, uuid

KAFKA_TOPIC_NAME_CONS = "cctv_vehicle_counts"   #topic
KAFKA_BOOTSTRAP_SERVERS_CONS = '127.0.0.1:9092'
SENSOR_NAME = "sensor_01"

if __name__ == "__main__":
    print("Kafka Producer Application Started ... ")
    kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTS

    message_list = []
```

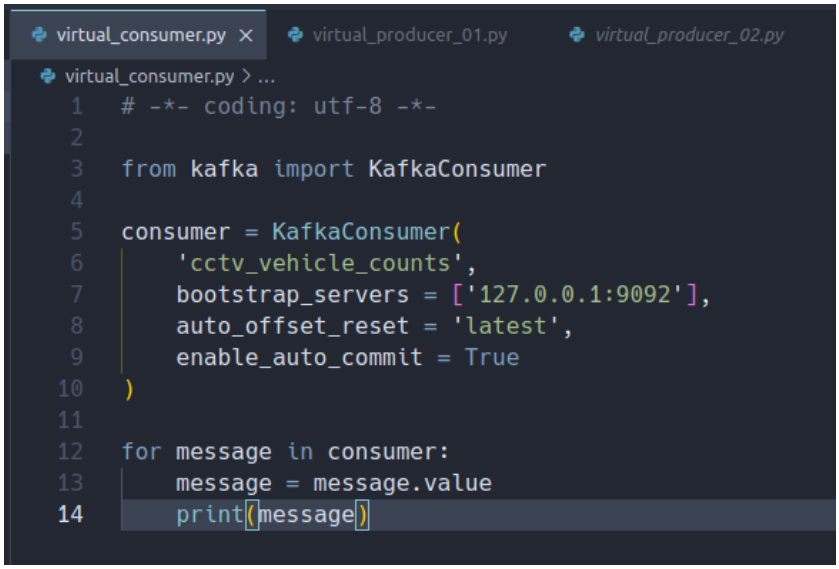
From the given producer script, correct this line to this so that it runs the lines inside the if statement.

```
if __name__ == "__main__":
    print("Kafka Producer Application Started ... ")
    kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTS
                                       value_serializer=lambda x: dump
```

Duplicate the producer script 9 times and change the SENSOR_NAME value inside the scripts to the appropriate value. Rename the files accordingly.

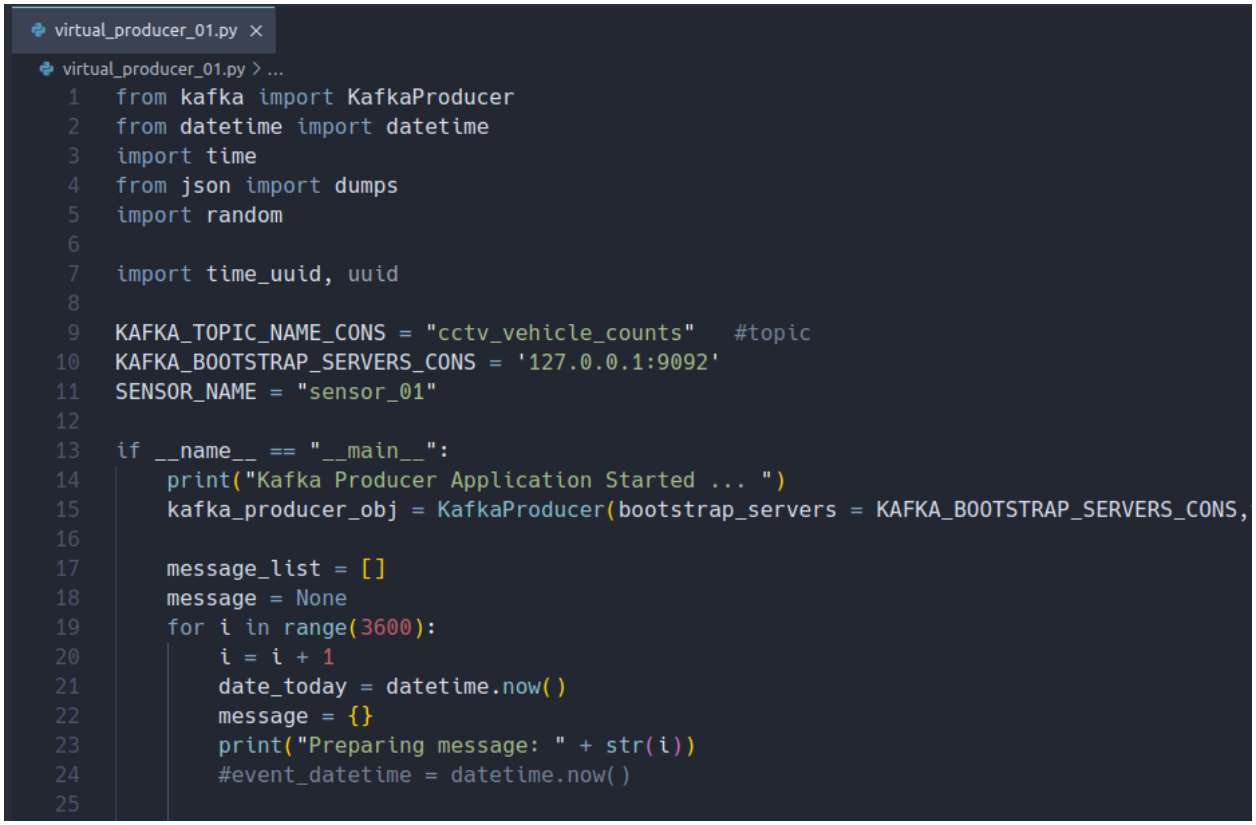


Snippet of the Consumer script



Snippets of the Producer scripts

Producer 1



Producer 2

virtual_producer_02.py

```python
from kafka import KafkaProducer
from datetime import datetime
import time
from json import dumps
import random

import time_uuid, uuid

KAFKA_TOPIC_NAME_CONS = "cctv_vehicle_counts"   #topic
KAFKA_BOOTSTRAP_SERVERS_CONS = '127.0.0.1:9092'
SENSOR_NAME = "sensor_02"

if __name__ == "__main__":
    print("Kafka Producer Application Started ... ")
    kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTSTRAP_SERVERS_CONS,va

    message_list = []
    message = None
    for i in range(3600):
        i = i + 1
        date_today = datetime.now()
        message = {}
        print("Preparing message: " + str(i))
        #event_datetime = datetime.now()
```

Producer 3

virtual_producer_03.py

```python
from kafka import KafkaProducer
from datetime import datetime
import time
from json import dumps
import random

import time_uuid, uuid

KAFKA_TOPIC_NAME_CONS = "cctv_vehicle_counts"   #topic
KAFKA_BOOTSTRAP_SERVERS_CONS = '127.0.0.1:9092'
SENSOR_NAME = "sensor_03"

if __name__ == "__main__":
    print("Kafka Producer Application Started ... ")
    kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTSTRAP_SERVERS_CONS,
                                       value_serializer=lambda x: dumps(x).encode('utf-8'

    message_list = []
    message = None
    for i in range(3600):
        i = i + 1
        date_today = datetime.now()
        message = {}
        print("Preparing message: " + str(i))
        #event_datetime = datetime.now()
```

Producer 4

virtual_producer_04.py

```python
from kafka import KafkaProducer
from datetime import datetime
import time
from json import dumps
import random

import time_uuid, uuid

KAFKA_TOPIC_NAME_CONS = "cctv_vehicle_counts"   #topic
KAFKA_BOOTSTRAP_SERVERS_CONS = '127.0.0.1:9092'
SENSOR_NAME = "sensor_04"

if __name__ == "__main__":
    print("Kafka Producer Application Started ... ")
    kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTSTRAP_SERVERS_CONS,
                                       value_serializer=lambda x: dumps(x).encode('utf-8'

    message_list = []
    message = None
    for i in range(3600):
        i = i + 1
        date_today = datetime.now()
        message = {}
        print("Preparing message: " + str(i))
        #event_datetime = datetime.now()
```

Producer 5

virtual_producer_05.py

```python
from kafka import KafkaProducer
from datetime import datetime
import time
from json import dumps
import random

import time_uuid, uuid

KAFKA_TOPIC_NAME_CONS = "cctv_vehicle_counts"   #topic
KAFKA_BOOTSTRAP_SERVERS_CONS = '127.0.0.1:9092'
SENSOR_NAME = "sensor_05"

if __name__ == "__main__":
    print("Kafka Producer Application Started ... ")
    kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTSTRAP_SERVERS_CONS,
                                       value_serializer=lambda x: dumps(x).encode('utf-8'

    message_list = []
    message = None
    for i in range(3600):
        i = i + 1
        date_today = datetime.now()
        message = {}
        print("Preparing message: " + str(i))
        #event_datetime = datetime.now()
```

Producer 6

virtual_producer_06.py

```python
from kafka import KafkaProducer
from datetime import datetime
import time
from json import dumps
import random

import time_uuid, uuid

KAFKA_TOPIC_NAME_CONS = "cctv_vehicle_counts"   #topic
KAFKA_BOOTSTRAP_SERVERS_CONS = '127.0.0.1:9092'
SENSOR_NAME = "sensor_06"

if __name__ == "__main__":
    print("Kafka Producer Application Started ... ")
    kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTSTRAP_SERVERS_CONS,
                                       value_serializer=lambda x: dumps(x).encode('utf-8'

    message_list = []
    message = None
    for i in range(3600):
        i = i + 1
        date_today = datetime.now()
        message = {}
        print("Preparing message: " + str(i))
```

Producer 7

virtual_producer_07.py

```python
from kafka import KafkaProducer
from datetime import datetime
import time
from json import dumps
import random

import time_uuid, uuid

KAFKA_TOPIC_NAME_CONS = "cctv_vehicle_counts"   #topic
KAFKA_BOOTSTRAP_SERVERS_CONS = '127.0.0.1:9092'
SENSOR_NAME = "sensor_07"

if __name__ == "__main__":
    print("Kafka Producer Application Started ... ")
    kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTSTRAP_SERVERS_CONS,
                                       value_serializer=lambda x: dumps(x).encode('utf-8

    message_list = []
    message = None
    for i in range(3600):
        i = i + 1
        date_today = datetime.now()
        message = {}
        print("Preparing message: " + str(i))
        #event_datetime = datetime.now()
```

Producer 8

virtual_producer_08.py

```python
from kafka import KafkaProducer
from datetime import datetime
import time
from json import dumps
import random

import time_uuid, uuid

KAFKA_TOPIC_NAME_CONS = "cctv_vehicle_counts"   #topic
KAFKA_BOOTSTRAP_SERVERS_CONS = '127.0.0.1:9092'
SENSOR_NAME = "sensor_08"

if __name__ == "__main__":
    print("Kafka Producer Application Started ... ")
    kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTSTRAP_SERVERS_CONS,
                                       value_serializer=lambda x: dumps(x).encode('utf-8'

    message_list = []
    message = None
    for i in range(3600):
        i = i + 1
        date_today = datetime.now()
        message = {}
        print("Preparing message: " + str(i))
        #event_datetime = datetime.now()
```

Producer 9

virtual_producer_09.py

```python
from kafka import KafkaProducer
from datetime import datetime
import time
from json import dumps
import random

import time_uuid, uuid

KAFKA_TOPIC_NAME_CONS = "cctv_vehicle_counts"   #topic
KAFKA_BOOTSTRAP_SERVERS_CONS = '127.0.0.1:9092'
SENSOR_NAME = "sensor_09"

if __name__ == "__main__":
    print("Kafka Producer Application Started ... ")
    kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTSTRAP_SERVERS_CONS,
                                       value_serializer=lambda x: dumps(x).encode('utf-8'

    message_list = []
    message = None
    for i in range(3600):
        i = i + 1
        date_today = datetime.now()
        message = {}
        print("Preparing message: " + str(i))
        #event_datetime = datetime.now()
```

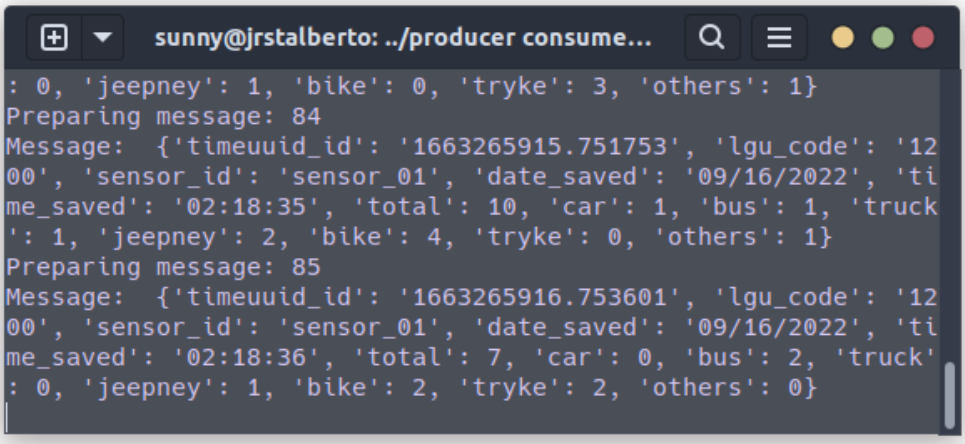
Producer 10

```python
from kafka import KafkaProducer
from datetime import datetime
import time
from json import dumps
import random

import time_uuid, uuid

KAFKA_TOPIC_NAME_CONS = "cctv_vehicle_counts"   #topic
KAFKA_BOOTSTRAP_SERVERS_CONS = '127.0.0.1:9092'
SENSOR_NAME = "sensor_10"

if __name__ == "__main__":
    print("Kafka Producer Application Started ... ")
    kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTSTRAP_SERVERS_CONS,
                                       value_serializer=lambda x: dumps(x).encode('utf-8'

    message_list = []
    message = None
    for i in range(3600):
        i = i + 1
        date_today = datetime.now()
        message = {}
        print("Preparing message: " + str(i))
        #event_datetime = datetime.now()
```

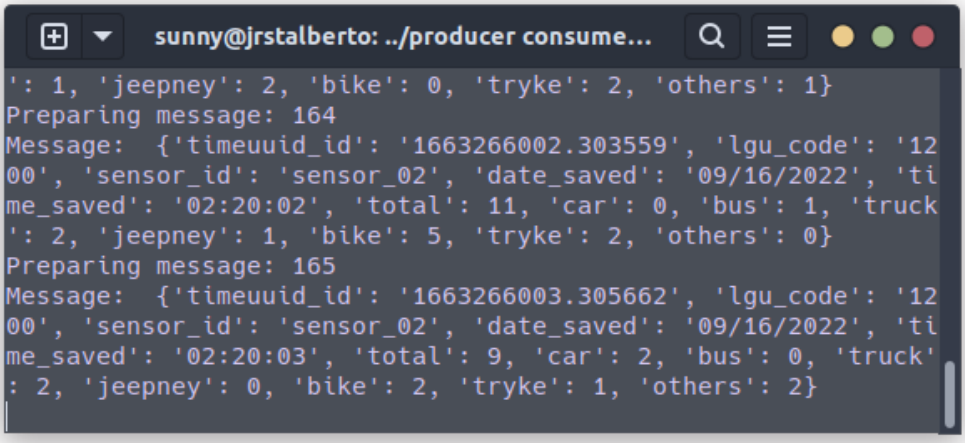
Running producers Simultaneously

Start 10 new terminal instances and run the python producer files simultaneously each with their own terminal instances.

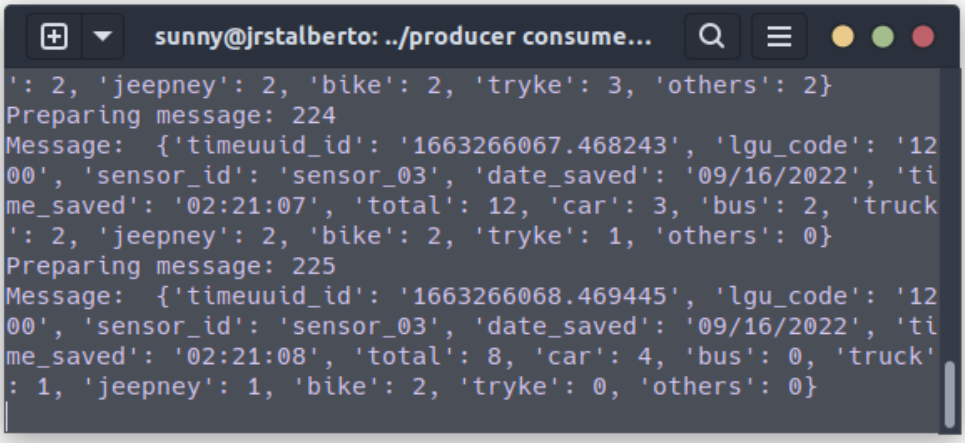
Producer 1



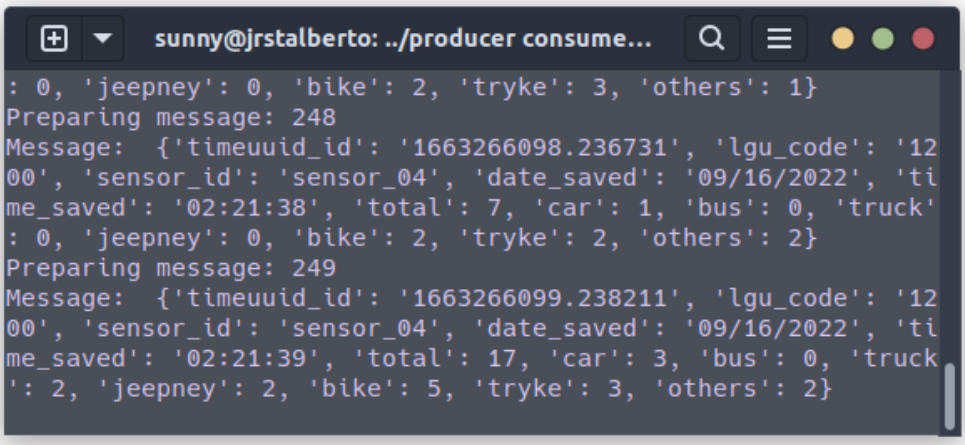
Producer 2



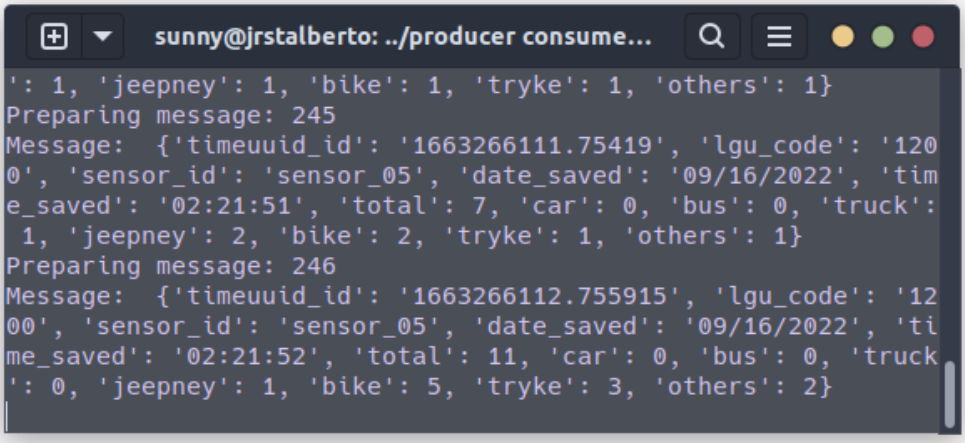
Producer 3



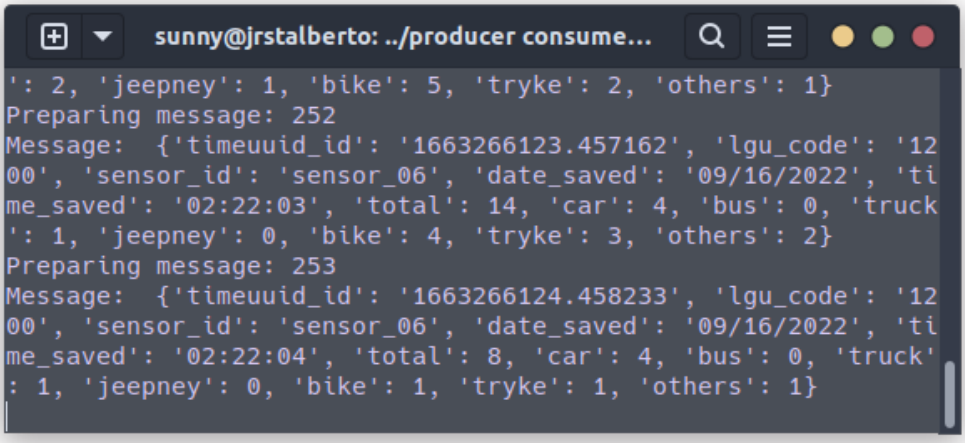
Producer 4



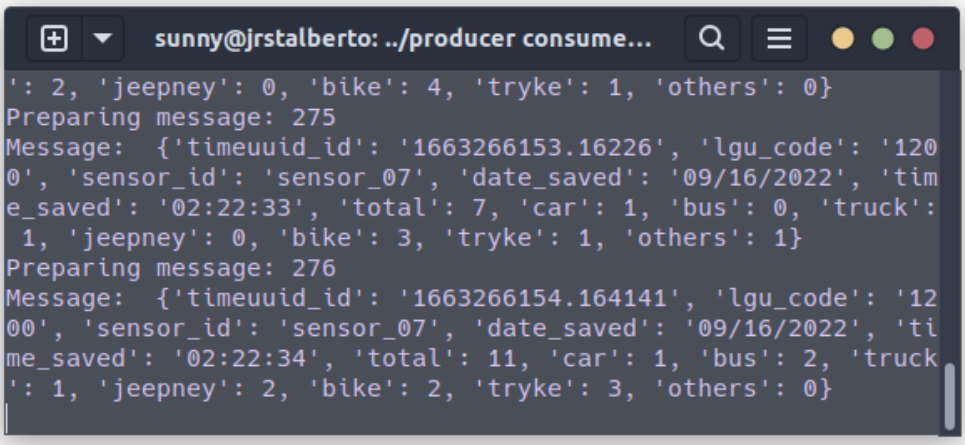
Producer 5



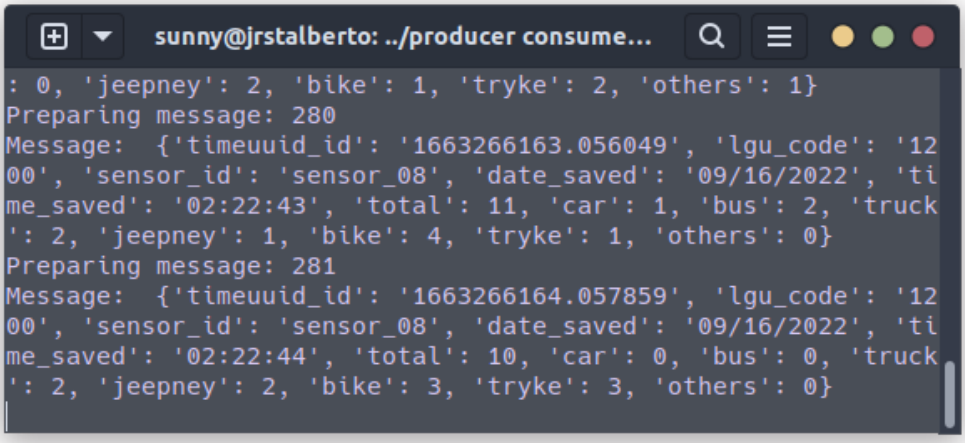
Producer 6



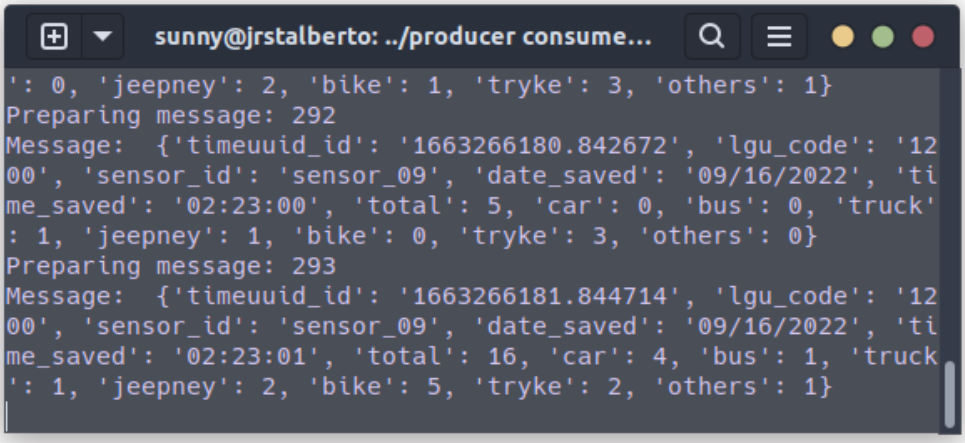
Producer 7



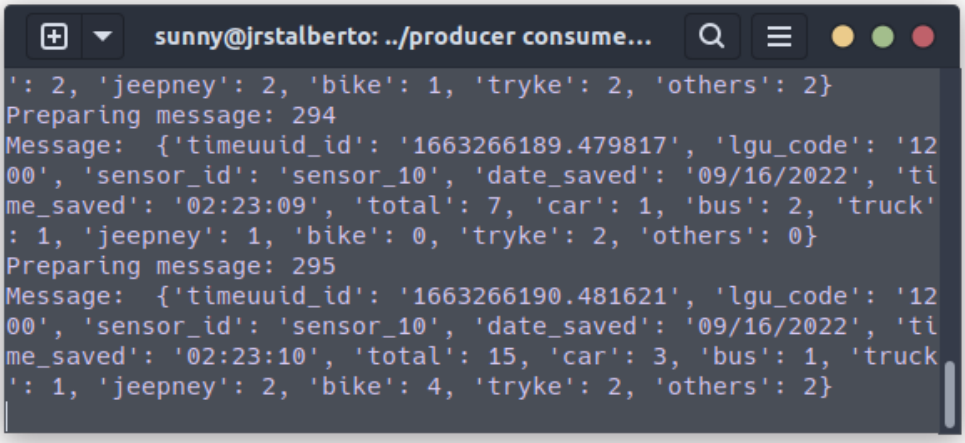
Producer 8



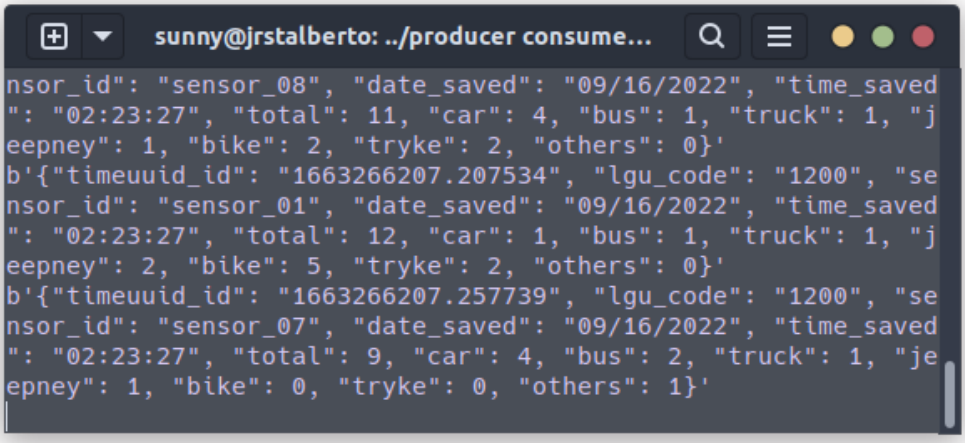
Producer 9

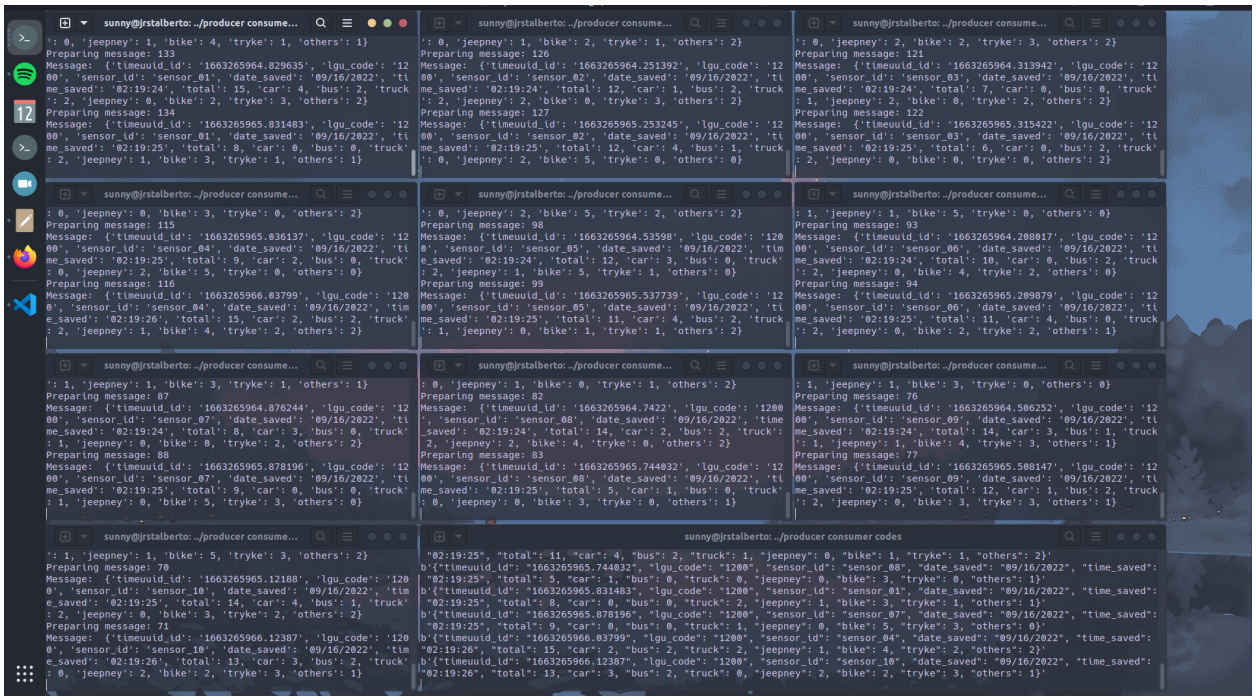


Producer 10



Consumer

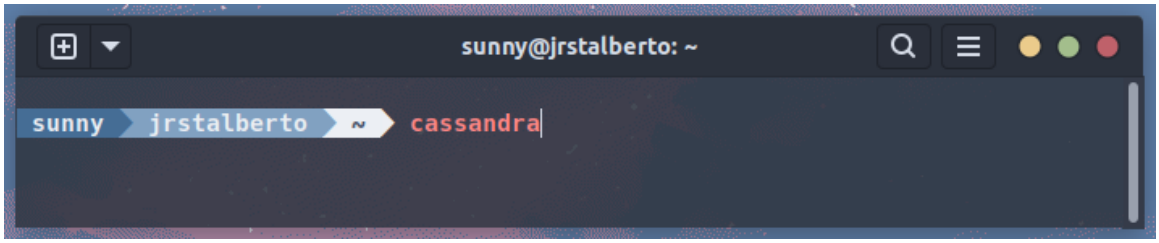

sunny@jrstalberto: ../producer consume...
nsor_id": "sensor_08", "date_saved": "09/16/2022", "time_saved
": "02:23:27", "total": 11, "car": 4, "bus": 1, "truck": 1, "j
eepney": 1, "bike": 2, "tryke": 2, "others": 0}'
b'{"timeuuid_id": "1663266207.207534", "lgu_code": "1200", "se
nsor_id": "sensor_01", "date_saved": "09/16/2022", "time_saved
": "02:23:27", "total": 12, "car": 1, "bus": 1, "truck": 1, "j
eepney": 2, "bike": 5, "tryke": 2, "others": 0}'
b'{"timeuuid_id": "1663266207.257739", "lgu_code": "1200", "se
nsor_id": "sensor_07", "date_saved": "09/16/2022", "time_saved
": "02:23:27", "total": 9, "car": 4, "bus": 2, "truck": 1, "je
epney": 1, "bike": 0, "tryke": 0, "others": 1}'



Initialize the Cassandra Environment

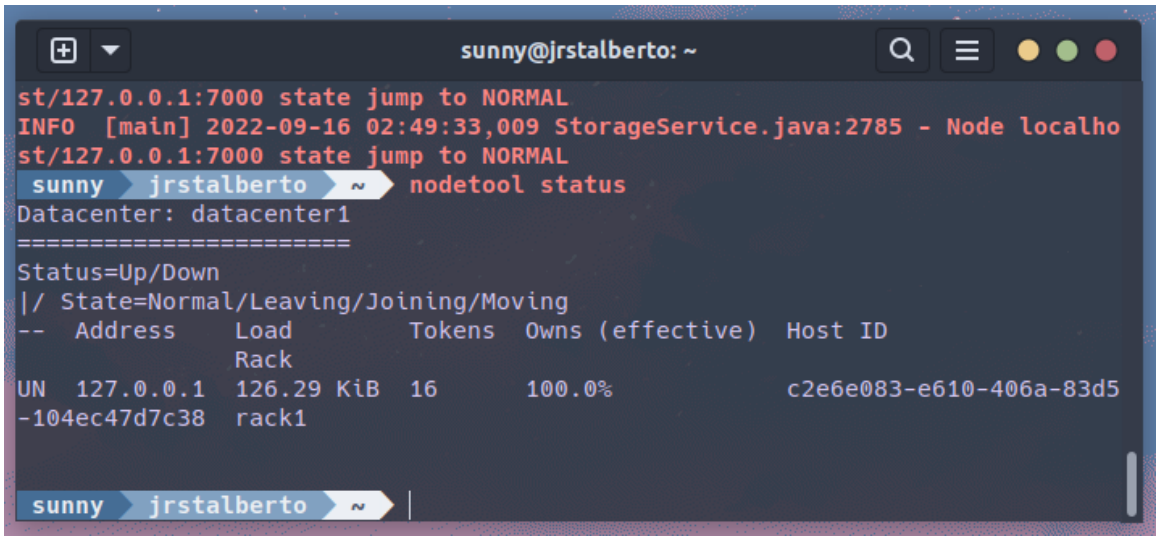
Start the Cassandra environment

```
cassandra
```



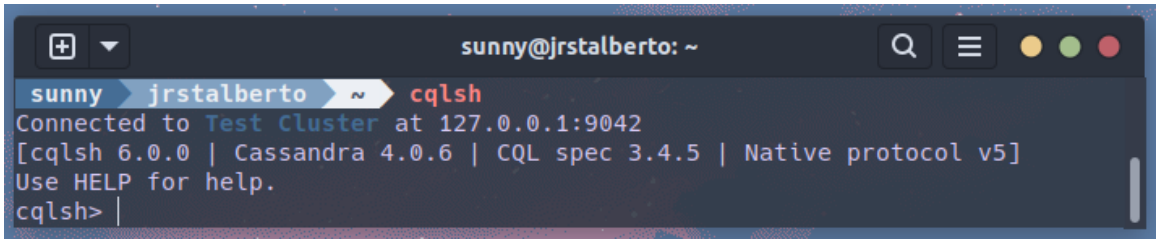
Verify if Cassandra started successfully

```
nodetool status
```



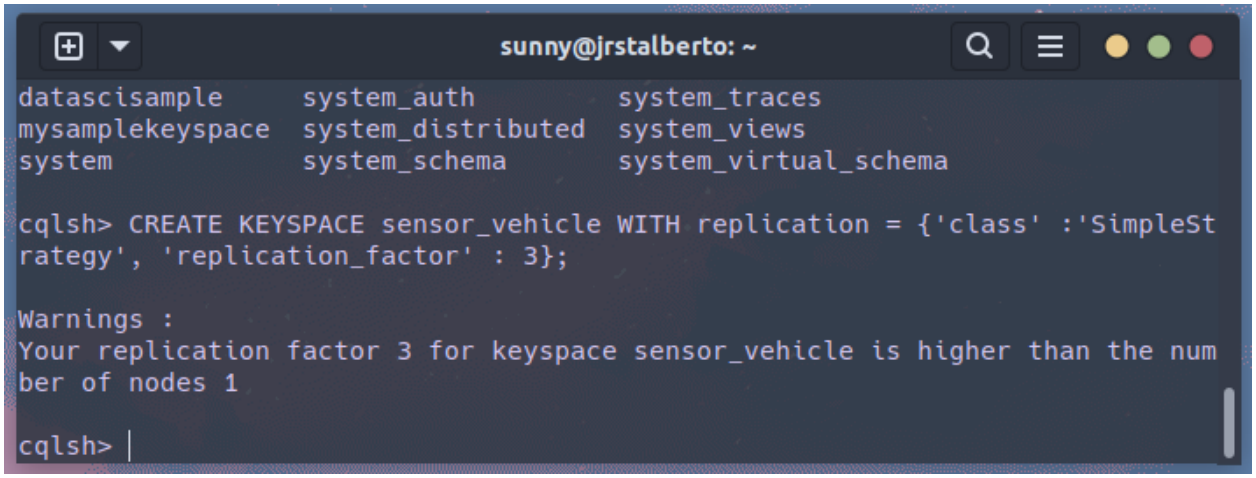
Connect to the database

```
cqlsh
```

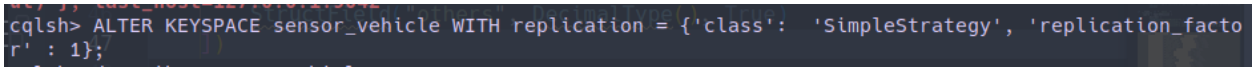


Create a new keyspace

```
CREATE KEYSPACE sensor_vehicle WITH replication = {'class'
:'SimpleStrategy', 'replication_factor' : 3};
```



Note: there were errors due to the differences of the replication factors of kafka topic and cassandra keyspace which was fixed by **altering the keyspace's replication factor** *to match the kafka topic's.*



Access the keyspace using

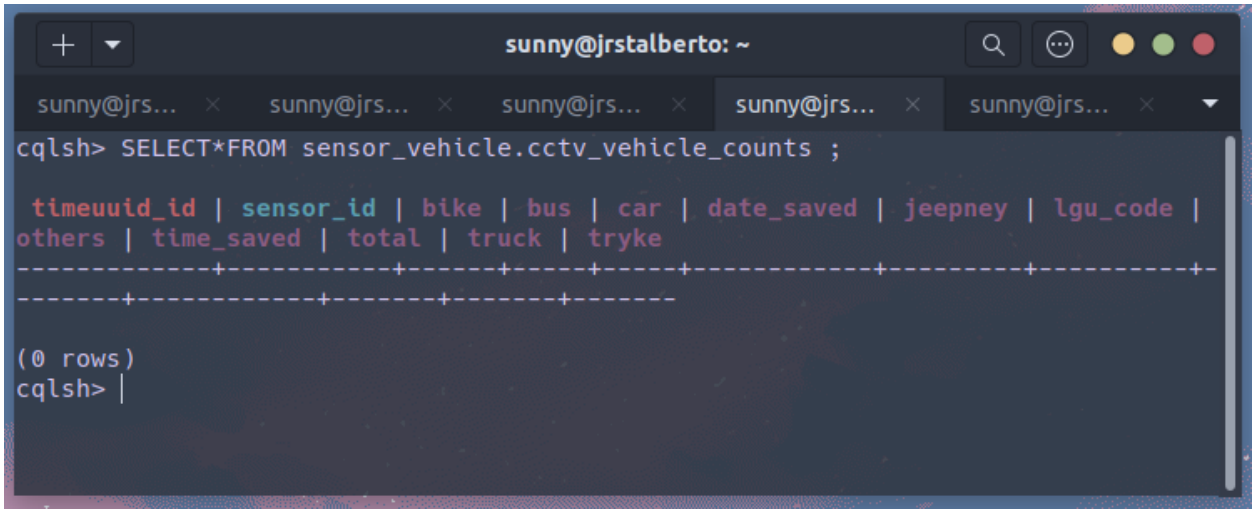
```
use sensor_vehicle
```

Create the table

```
CREATE TABLE cctv_vehicle_counts(
    ... timeuuid_id varchar,
    ... lgu_code varchar,
    ... sensor_id varchar,
    ... date_saved varchar,
    ... time_saved varchar,
    ... total int,
    ... car int,
    ... bus int,
    ... truck int,
    ... jeepney int,
    ... bike int,
    ... tryke int,
    ... others int,
    ... PRIMARY KEY(timeuuid_id, sensor_id)
    ... );
```

Check if the table was created successfully

```
SELECT*FROM sensor_vehicle.cctv_vehicle_counts;
```



Initialize the Spark Environment

Prepare the code for the stream handler

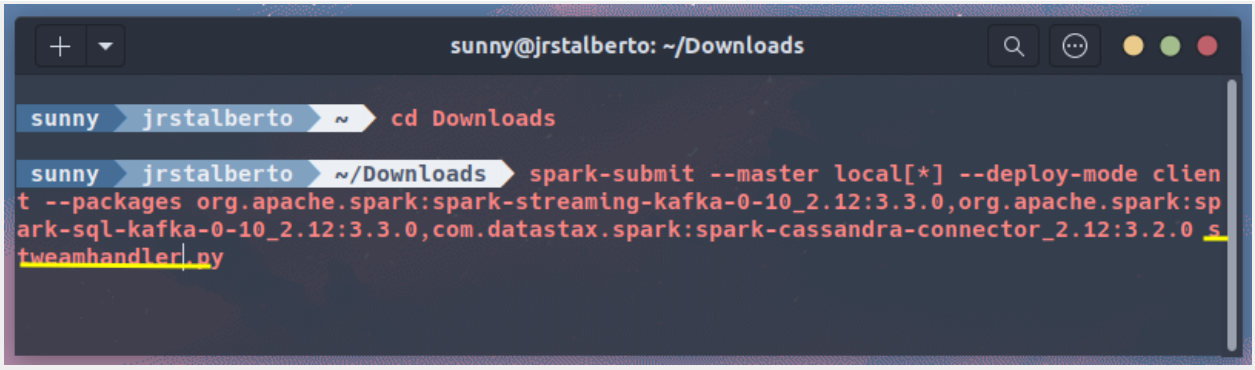
stweamhandler.py

```python
import findspark
findspark.init()

from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *

cassandra_host_name = "localhost"
cassandra_port_no = "9042"
cassandra_keyspace_name = "sensor_vehicle"
cassandra_table_name = "cctv_vehicle_counts"

def save_to_cassandra(current_df, epoch_id):
    print("Printing epoch_id: ")
    print(epoch_id)

    print("Printing before Cassandra Table save: " + str(epoch_id))
    current_df \
        .write \
        .format("org.apache.spark.sql.cassandra") \
        .mode("append") \
        .options(table=cassandra_table_name, keyspace=cassandra_keyspace_name) \
        .save()

if __name__ == "__main__":
    spark = SparkSession \
        .builder \
        .appName("StreamHandler") \
        .config('spark.cassandra.connection.host', cassandra_host_name) \
        .config('spark.cassandra.connection.port', cassandra_port_no) \
        .getOrCreate()
```

```python

    schema = StructType([
        StructField("timeuuid_id", StringType(), True),
        StructField("lgu_code", StringType(), True),
        StructField("sensor_id", StringType(), True),
        StructField("date_saved", StringType(), True),
        StructField("time_saved", StringType(), True),
        StructField("total", DecimalType(), True),
        StructField("car", DecimalType(), True),
        StructField("bus", DecimalType(), True),
        StructField("truck", DecimalType(), True),
        StructField("jeepney", DecimalType(), True),
        StructField("bike", DecimalType(), True),
        StructField("tryke", DecimalType(), True),
        StructField("others", DecimalType(), True)
    ])

    lines = spark \
        .readStream \
        .format("kafka") \
        .option("kafka.bootstrap.servers", "localhost:9092") \
        .option("subscribe", "cctv_vehicle_counts") \
        .option("startingOffsets", "latest") \
        .load() \
        .select(from_json(col("value").cast("string"), schema).alias("parsed_value")).select(col("parsed_valu

    query = lines \
        .writeStream \
        .trigger(processingTime='15 seconds') \
        .outputMode("update") \
        .foreachBatch(save_to_cassandra) \
        .start()

    query.awaitTermination()
```

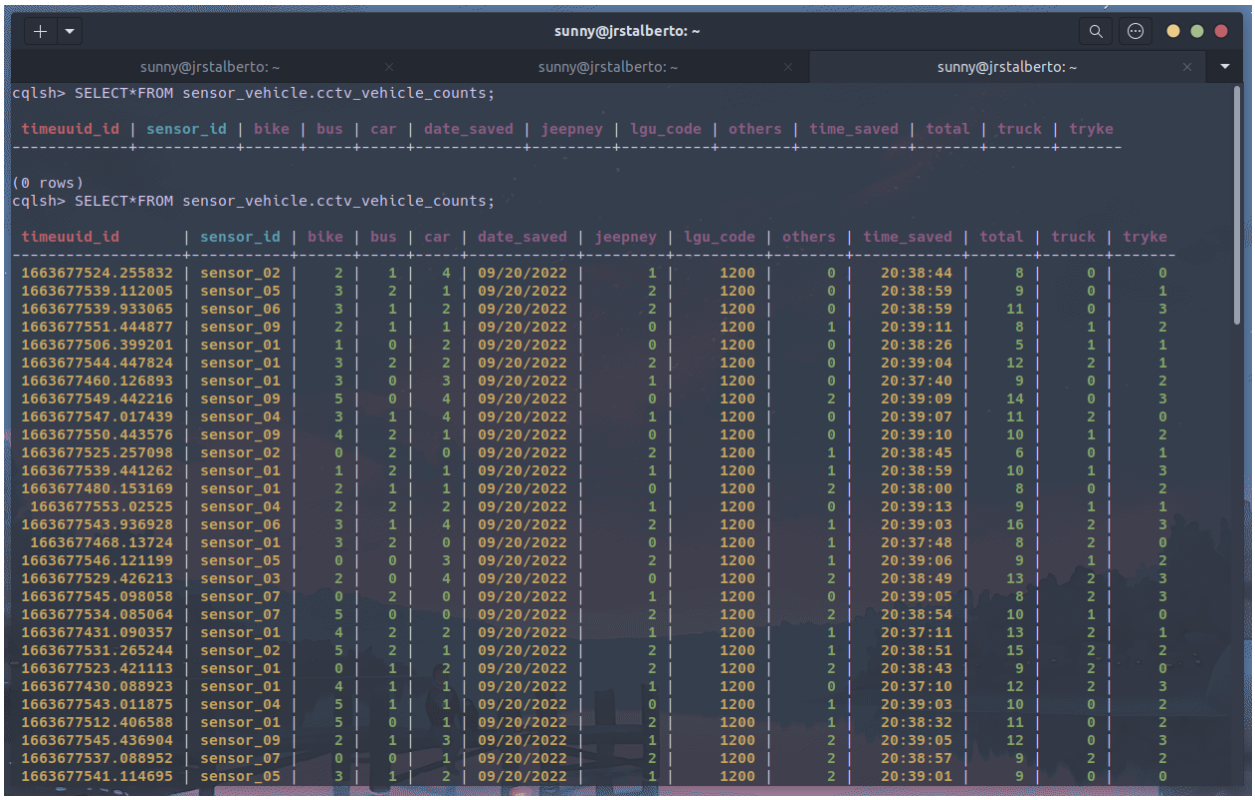
Save the file then navigate to the directory where the stream handler was saved, then run the command taking note of the name of the file at the end

```
spark-submit --master local[*] --deploy-mode client --packages org.apache.spark:spark-streaming-kafka-0-10_2.12:3.3.0,org.apache.spark:spark-sql-kafka-0-10_2.12:3.3.0,com.datastax.spark:spark-cassandra-connector_2.12:3.2.0 stweamhandler.py
```



Send a query at the cassandra terminal to see if the data are successfully saved to the cassandra table.

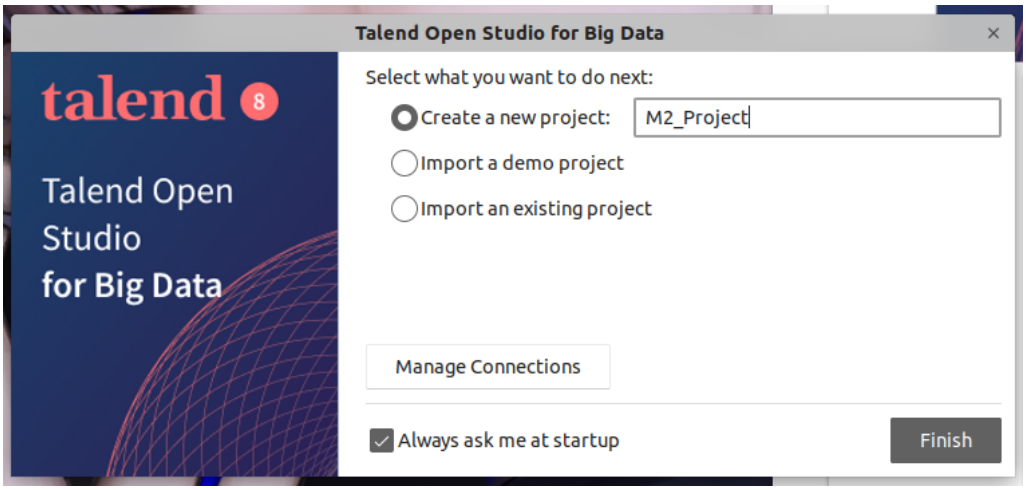
```
SELECT*FROM sensor_vehicle.cctv_vehicle_counts;
```



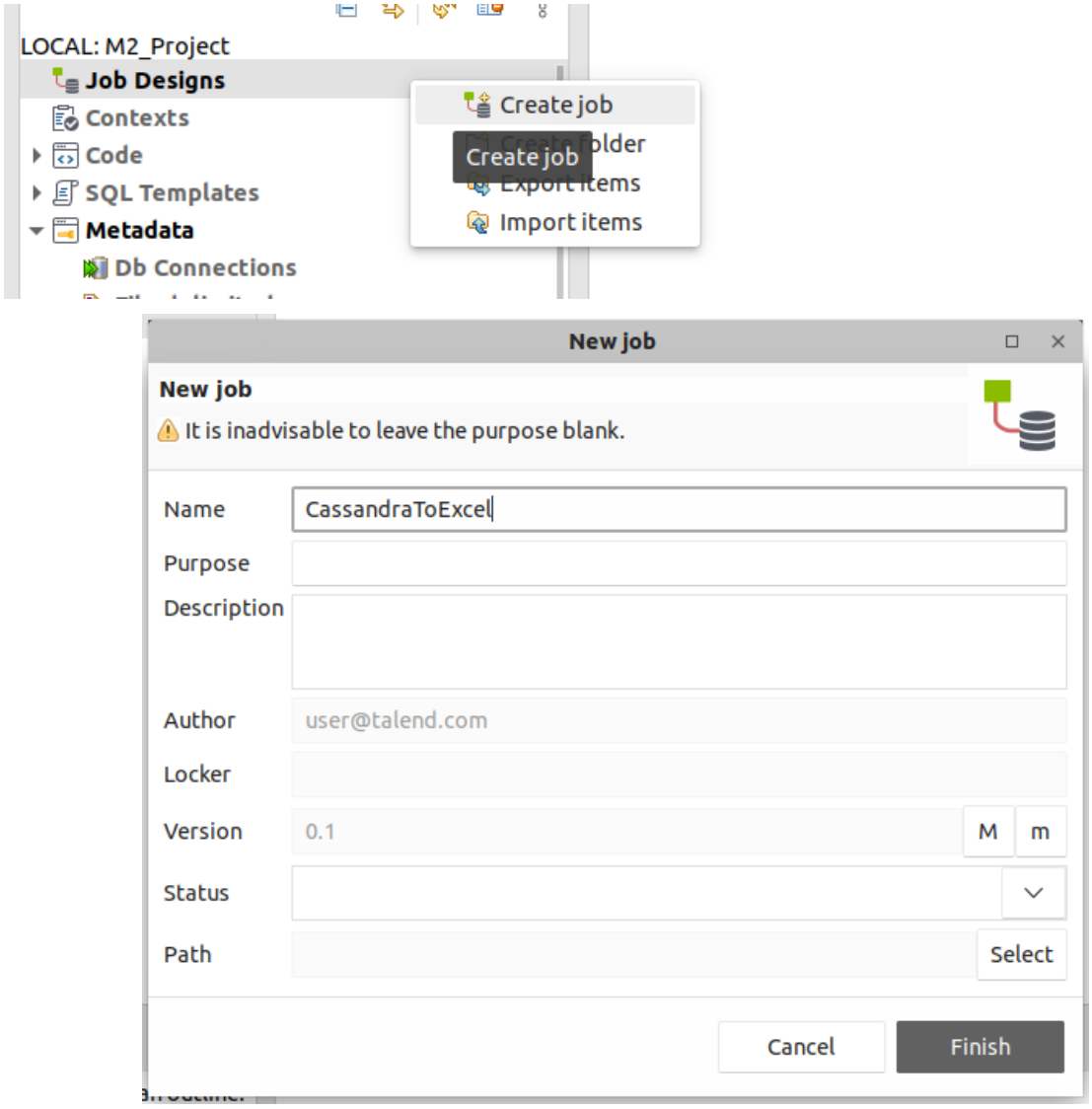
Start Cassandra

```
sunny  jrstalberto  ~  cassandra

sunny  jrstalberto  ~  CompilerOracle: dontinline org/apache/cassandra/db/Co
lumns$Serializer.deserializeLargeSubset (Lorg/apache/cassandra/io/util/DataInput
Plus;Lorg/apache/cassandra/db/Columns;I)Lorg/apache/cassandra/db/Columns;
CompilerOracle: dontinline org/apache/cassandra/db/Columns$Serializer.serializeL
argeSubset (Ljava/util/Collection;ILorg/apache/cassandra/db/Columns;ILorg/apache
/cassandra/io/util/DataOutputPlus;)V
CompilerOracle: dontinline org/apache/cassandra/db/Columns$Serializer.serializeL
argeSubsetSize (Ljava/util/Collection;ILorg/apache/cassandra/db/Columns;I)I
CompilerOracle: dontinline org/apache/cassandra/db/commitlog/AbstractCommitLogSe
gmentManager.advanceAllocatingFrom (Lorg/apache/cassandra/db/commitlog/CommitLog
Segment;)V
CompilerOracle: dontinline org/apache/cassandra/db/transform/BaseIterator.tryGet
MoreContents ()Z
CompilerOracle: dontinline org/apache/cassandra/db/transform/StoppingTransformat
ion.stop ()V
CompilerOracle: dontinline org/apache/cassandra/db/transform/StoppingTransformat
ion.stopInPartition ()V
CompilerOracle: dontinline org/apache/cassandra/io/util/BufferedDataOutputStream
Plus.doFlush (I)V
CompilerOracle: dontinline org/apache/cassandra/io/util/BufferedDataOutputStream
Plus.writeSlow (JI)V
CompilerOracle: dontinline org/apache/cassandra/io/util/RebufferingInputStream.r
eadPrimitiveSlowly (I)J
CompilerOracle: exclude org/apache/cassandra/utils/JVMStabilityInspector.forceHe
apSpaceOomMaybe (Ljava/lang/OutOfMemoryError;)V
```

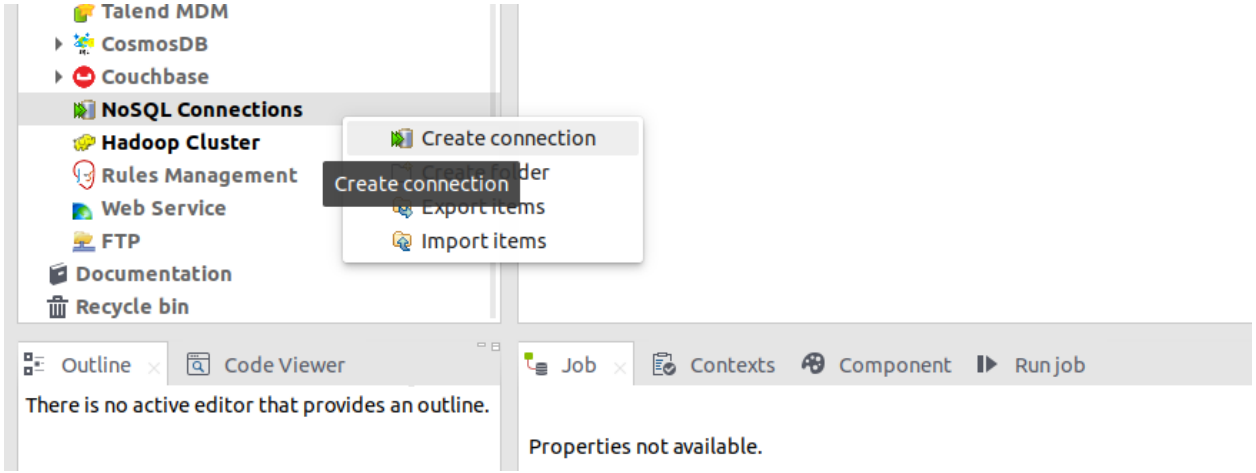
Create a new project in Talend Open Studio for Big Data

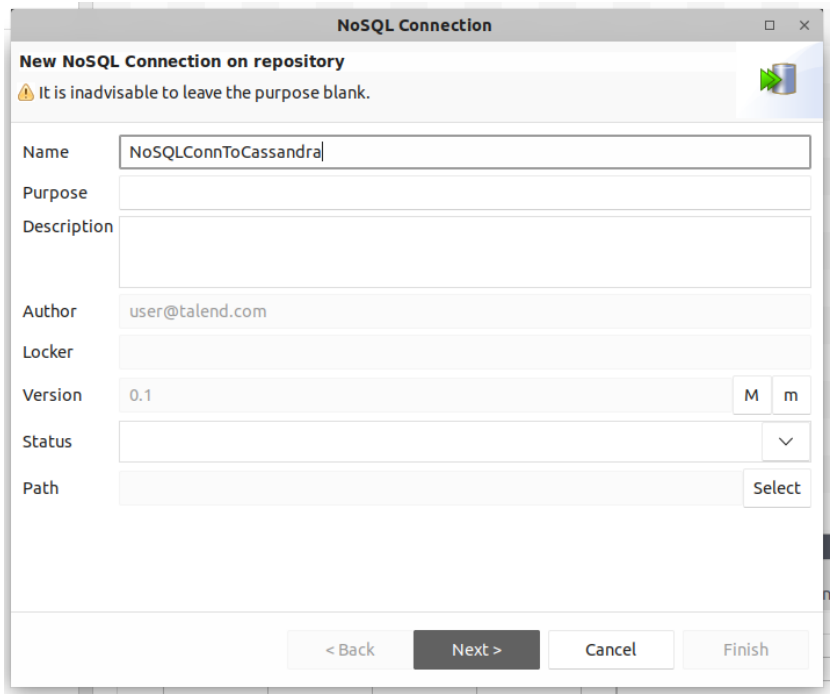


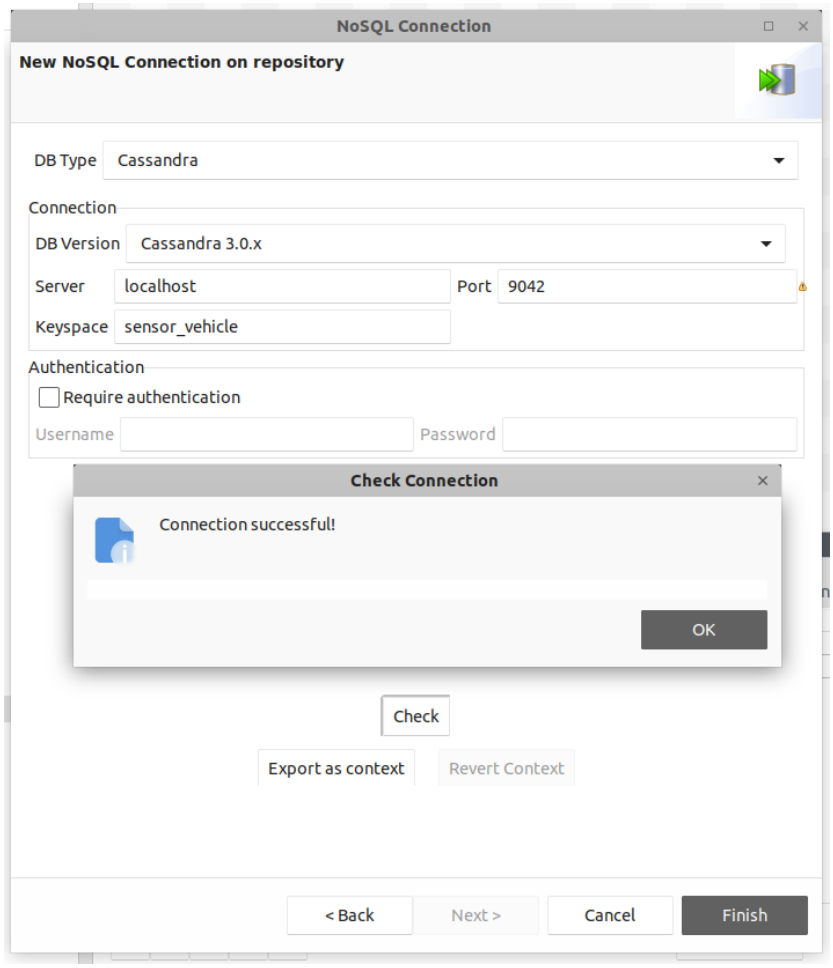
Create a new job.

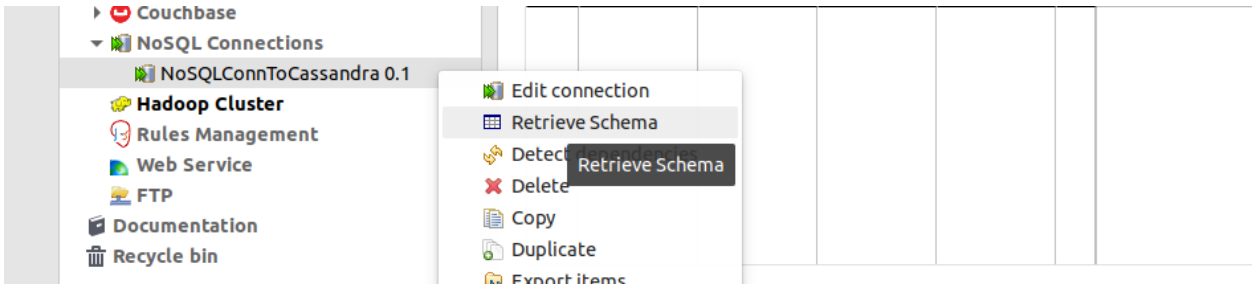


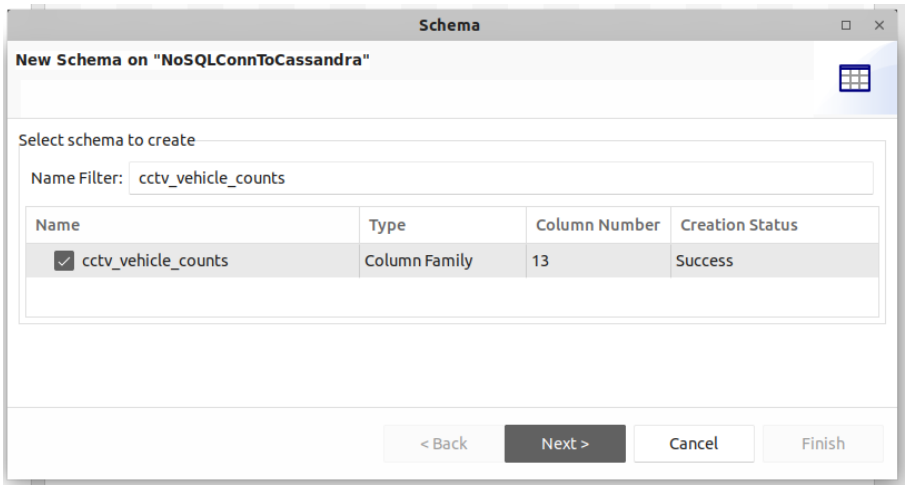
Create a new NoSQL connection.

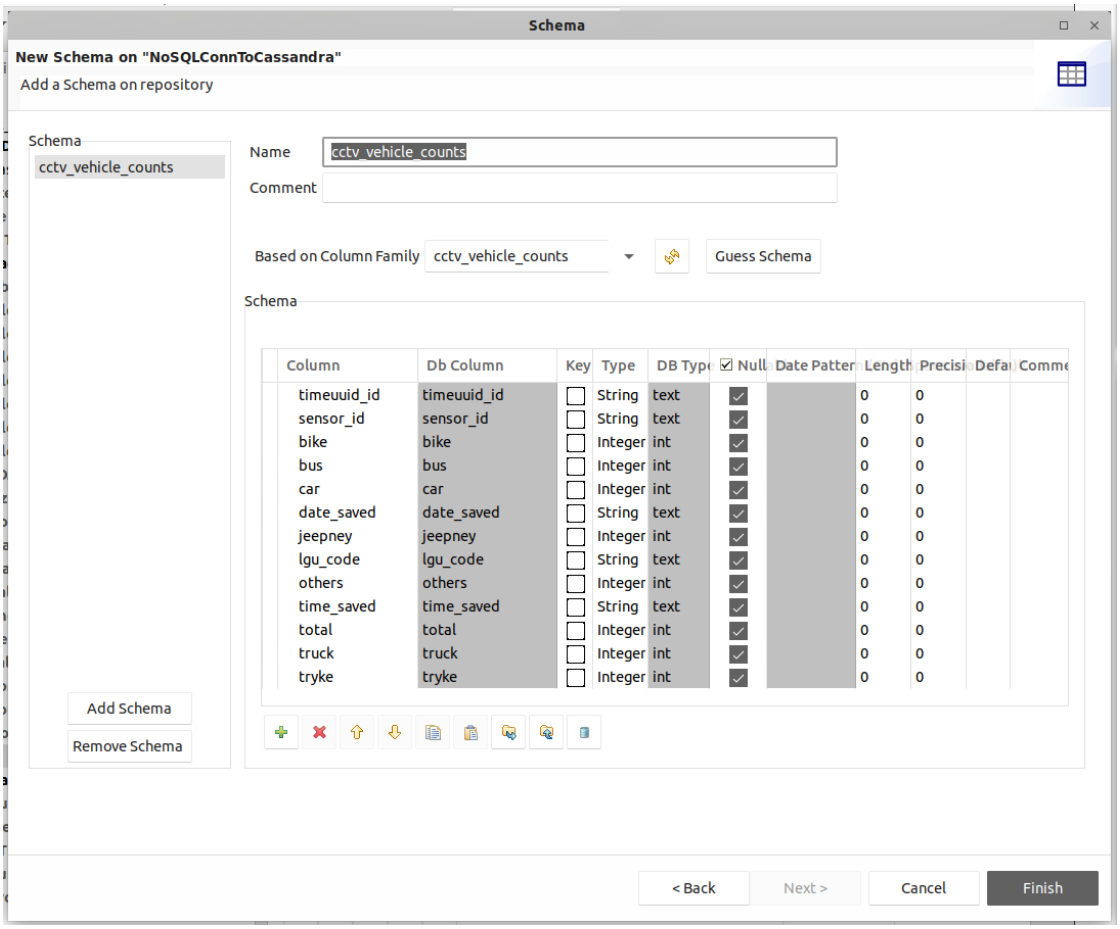


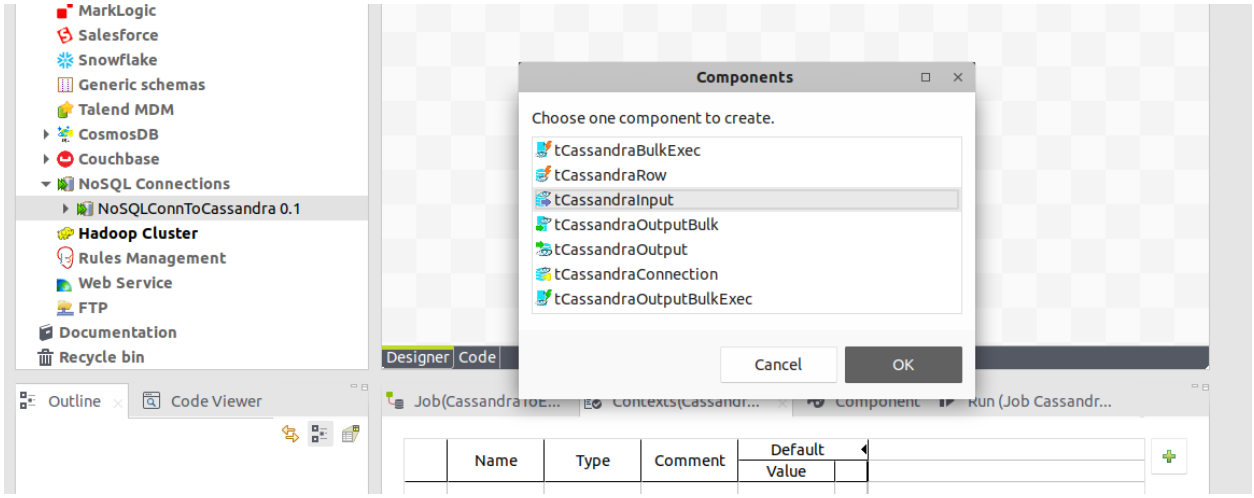

NoSQL Connection
New NoSQL Connection on repository
It is inadvisable to leave the purpose blank.
Name
NoSQLConnToCassandra
Purpose
Description
Author
user@talend.com
Locker
Version
0.1
M
m
Status
Path
Select
< Back
Next >
Cancel
Finish

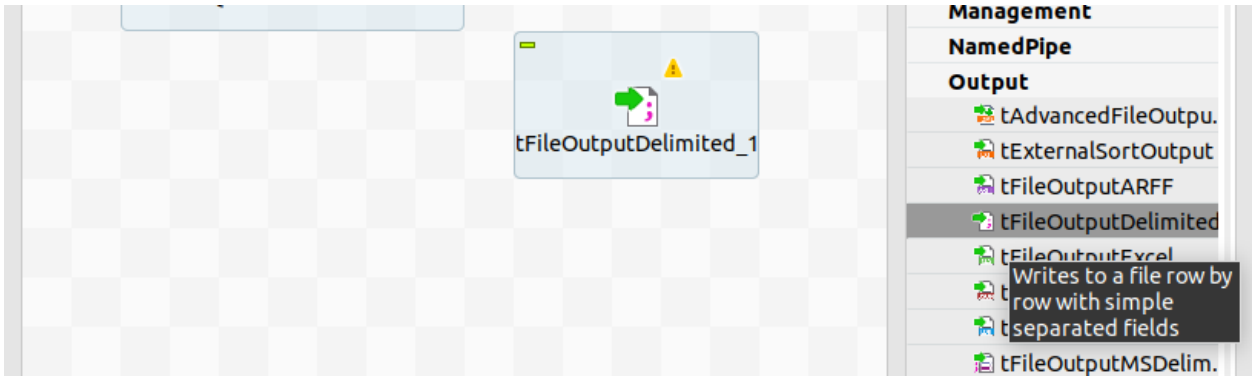

NoSQL Connection
New NoSQL Connection on repository
DB Type
Cassandra
Connection
DB Version
Cassandra 3.0.x
Server
localhost
Port
9042
Keyspace
sensor_vehicle
Authentication
Require authentication
Username
Password
Check Connection
Connection successful!
OK
Check
Export as context
Revert Context
< Back
Next >
Cancel
Finish

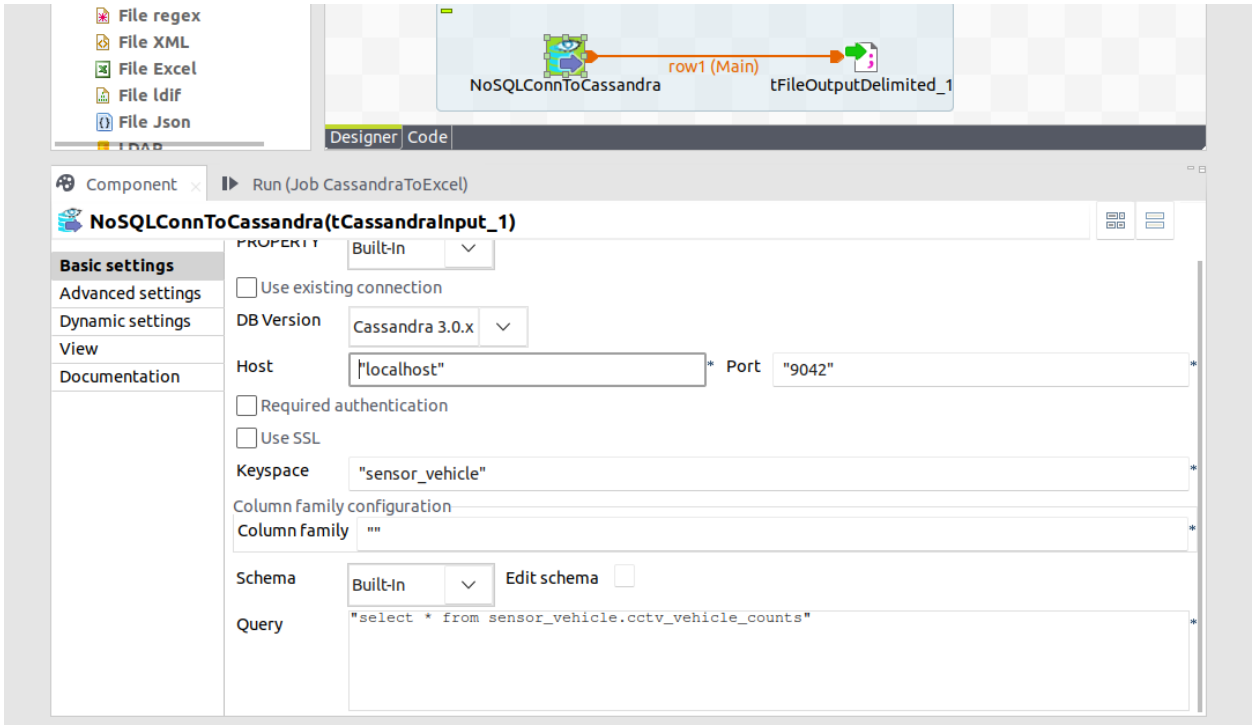

Couchbase
NoSQL Connections
NoSQLConnToCassandra 0.1
Hadoop Cluster
Rules Management
Web Service
FTP
Documentation
Recycle bin
Edit connection
Retrieve Schema
Delete
Copy
Duplicate
Retrieve Schema

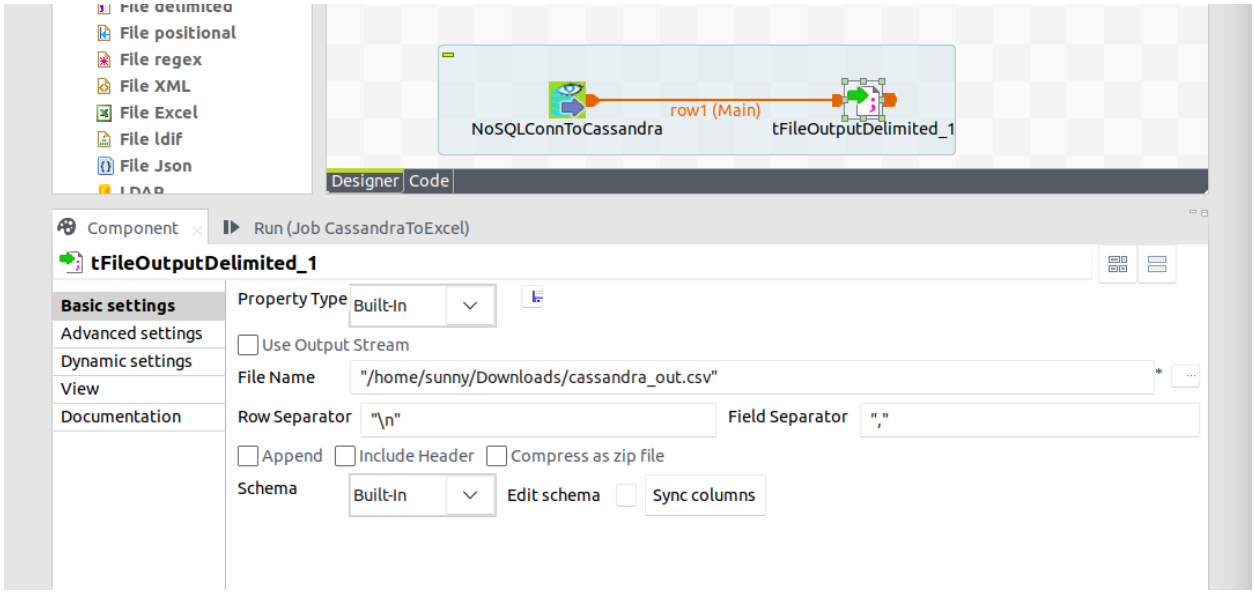

Schema
New Schema on "NoSQLConnToCassandra"
Select schema to create
Name Filter: cctv_vehicle_counts
Name
Type
Column Number
Creation Status
cctv_vehicle_counts
Column Family
13
Success
< Back
Next >
Cancel
Finish

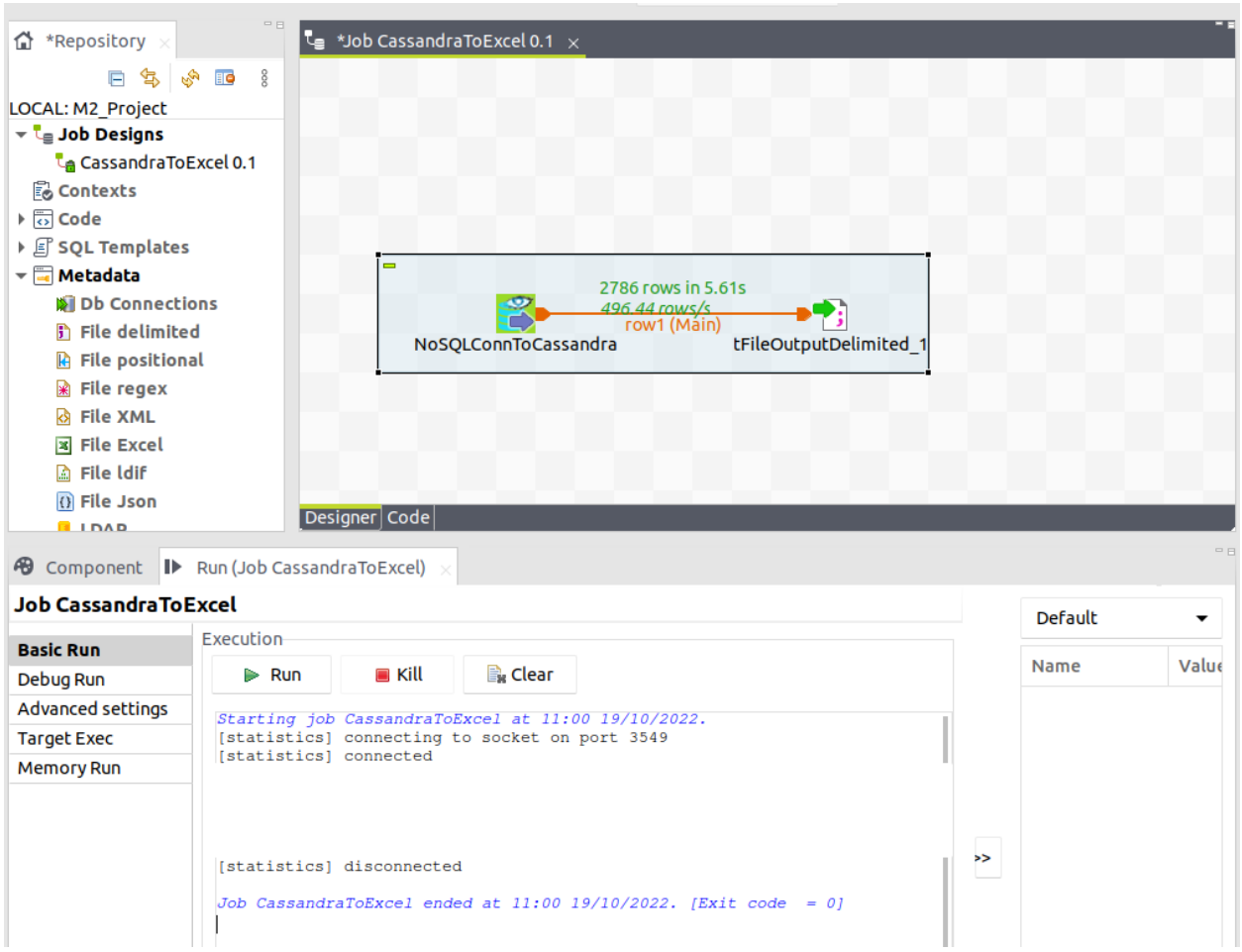

Schema
New Schema on "NoSQLConnToCassandra"
Add a Schema on repository
Schema
cctv_vehicle_counts
Name
cctv_vehicle_counts
Comment
Based on Column Family
cctv_vehicle_counts
Guess Schema
Schema
Column
Db Column
Key
Type
DB Type
Null
Date Patter
Length
Precisi
Defa
Comme
timeuuid_id
sensor_id
bike
bus
car
date_saved
jeepney
lgu_code
others
time_saved
total
truck
tryke
String
Integer
text
int
0
Add Schema
Remove Schema
< Back
Next >
Cancel
Finish

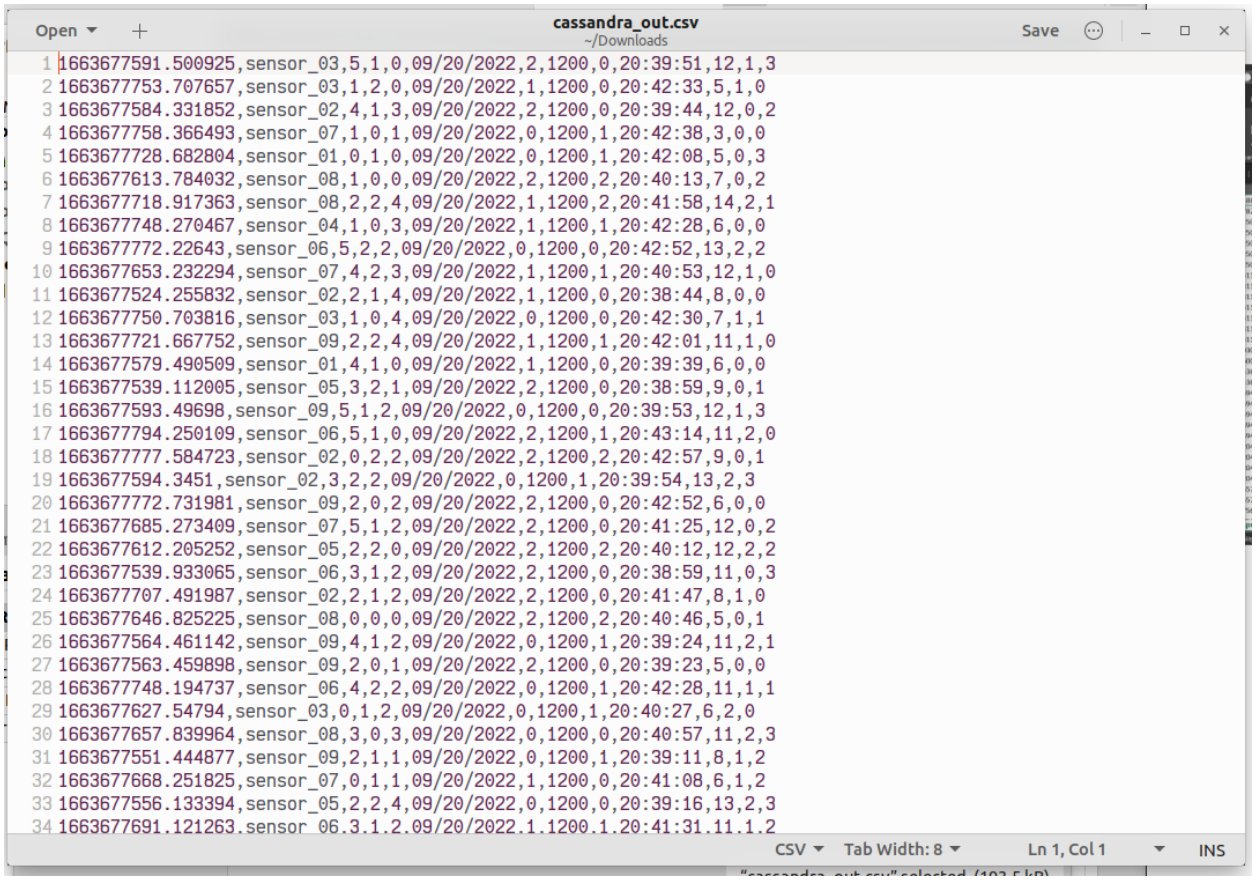

MarkLogic
Salesforce
Snowflake
Generic schemas
Talend MDM
CosmosDB
Couchbase
NoSQL Connections
NoSQLConnToCassandra 0.1
Hadoop Cluster
Rules Management
Web Service
FTP
Documentation
Recycle bin
Components
Choose one component to create.
tCassandraBulkExec
tCassandraRow
tCassandraInput
tCassandraOutputBulk
tCassandraOutput
tCassandraConnection
tCassandraOutputBulkExec
Cancel
OK
Designer
Code
Outline
Code Viewer
Name
Type
Comment
Default Value


tFileOutputDelimited_1
Management
NamedPipe
Output
tAdvancedFileOutpu.
tExternalSortOutput
tFileOutputARFF
tFileOutputDelimited
Writes to a file row by row with simple separated fields
tFileOutputMSDelim.


File regex
File XML
File Excel
File ldif
File Json
NoSQLConnToCassandra
row1 (Main)
tFileOutputDelimited_1
Designer
Code
Component
Run (Job CassandraToExcel)
NoSQLConnToCassandra(tCassandraInput_1)
Basic settings
Advanced settings
Dynamic settings
View
Documentation
Built-In
Use existing connection
DB Version
Cassandra 3.0.x
Host
"localhost"
Port
"9042"
Required authentication
Use SSL
Keyspace
"sensor_vehicle"
Column family configuration
Column family
""
Schema
Built-In
Edit schema
Query
"select * from sensor_vehicle.cctv_vehicle_counts"


File positional
File regex
File XML
File Excel
File ldif
File Json
NoSQLConnToCassandra
row1 (Main)
tFileOutputDelimited_1
Designer
Code
Component
Run (Job CassandraToExcel)
tFileOutputDelimited_1
Basic settings
Advanced settings
Dynamic settings
View
Documentation
Property Type
Built-In
Use Output Stream
File Name
"/home/sunny/Downloads/cassandra_out.csv"
Row Separator
"\n"
Field Separator
","
Append
Include Header
Compress as zip file
Schema
Built-In
Edit schema
Sync columns


*Repository
LOCAL: M2_Project
Job Designs
CassandraToExcel 0.1
Contexts
Code
SQL Templates
Metadata
Db Connections
File delimited
File positional
File regex
File XML
File Excel
File ldif
File Json
*Job CassandraToExcel 0.1
2786 rows in 5.61s
496.44 rows/s
row1 (Main)
NoSQLConnToCassandra
tFileOutputDelimited_1
Designer
Code
Component
Run (Job CassandraToExcel)
Job CassandraToExcel
Basic Run
Debug Run
Advanced settings
Target Exec
Memory Run
Execution
Run
Kill
Clear
Starting job CassandraToExcel at 11:00 19/10/2022.
[statistics] connecting to socket on port 3549
[statistics] connected
[statistics] disconnected
Job CassandraToExcel ended at 11:00 19/10/2022. [Exit code = 0]
Default
Name
Value



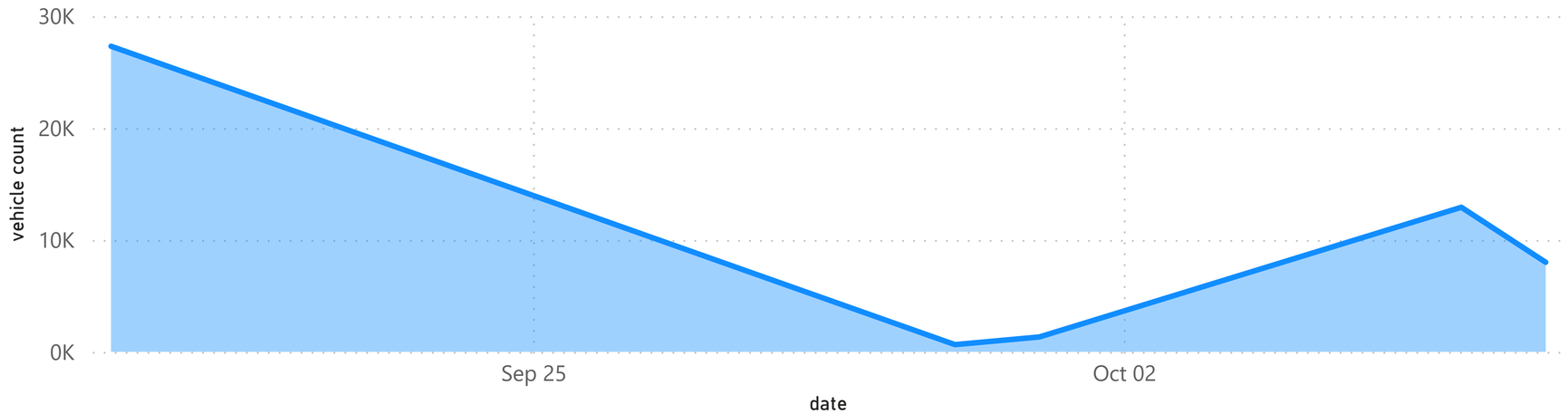
File successfully imported to CSV from cassandra keyspace using Talend Big Data.

Quick summary of vehicle sensor counts

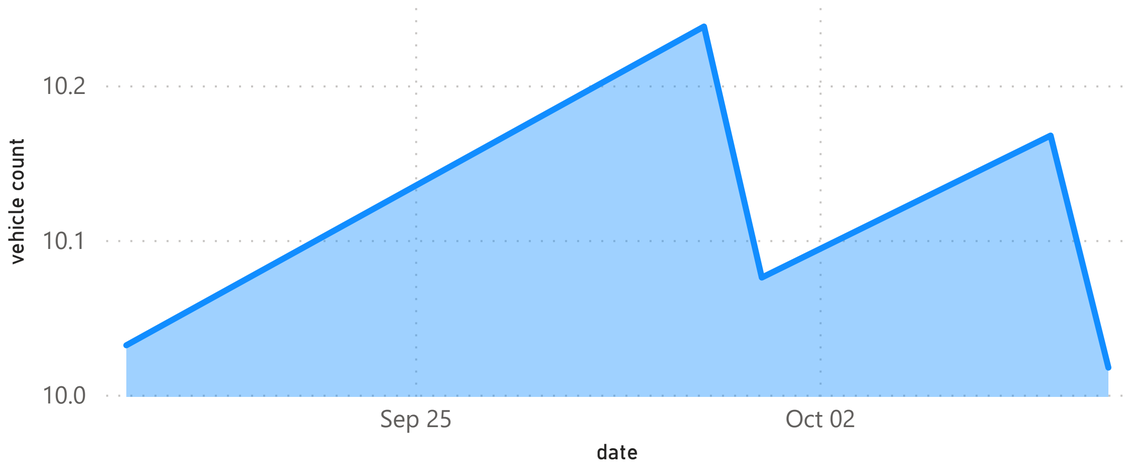
50219

Total vehicles counted

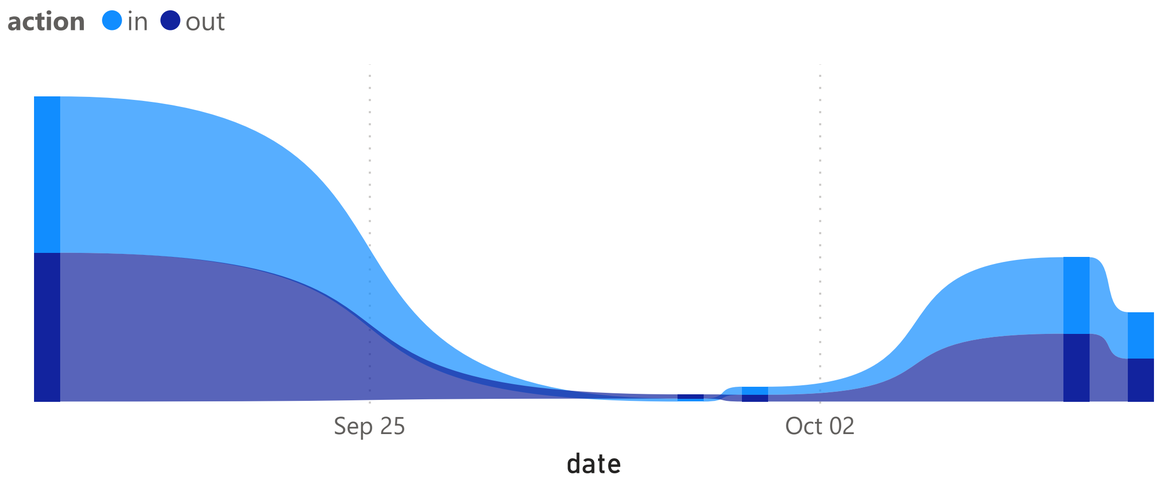
Vehicle counts over time



Average vehicle count per day



Vehicles counted by action over time

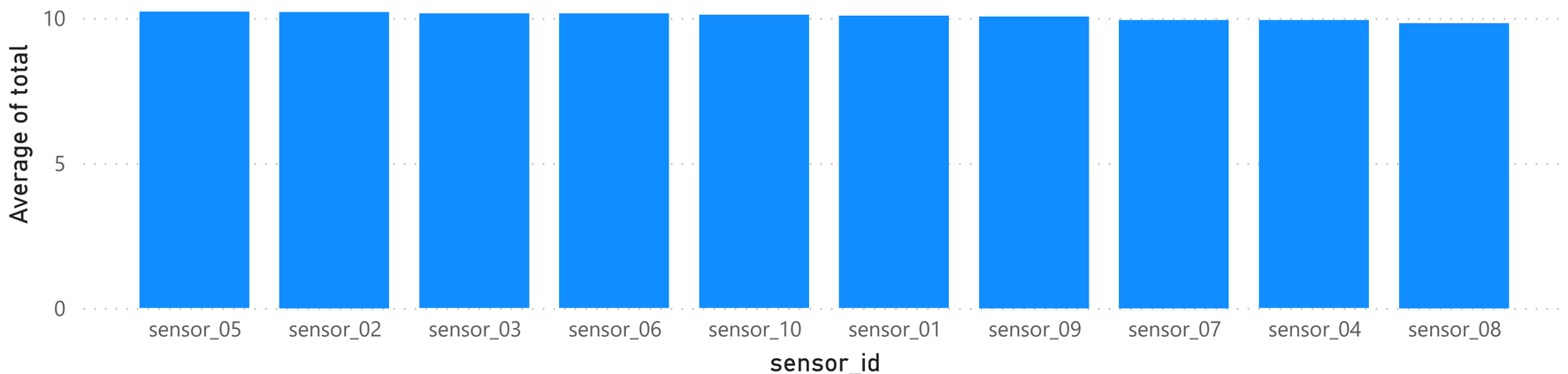


Quick summary of vehicle sensor counts

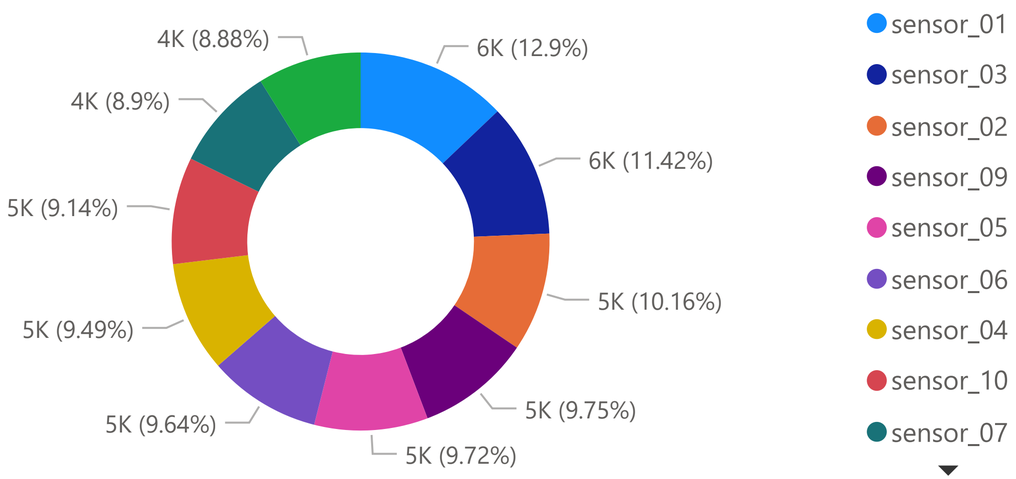
50219

Total vehicles counted

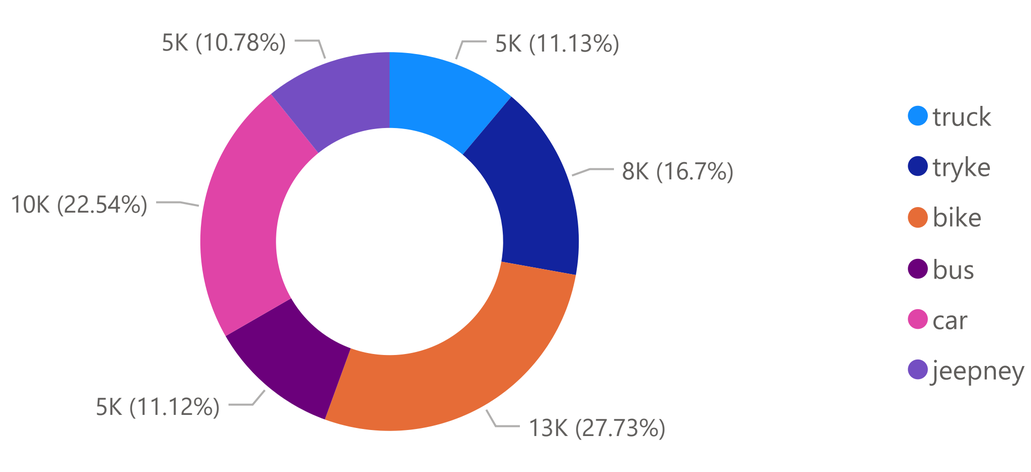
Average vehicle count per sensor



Total vehicle count per sensor



Vehicle type count

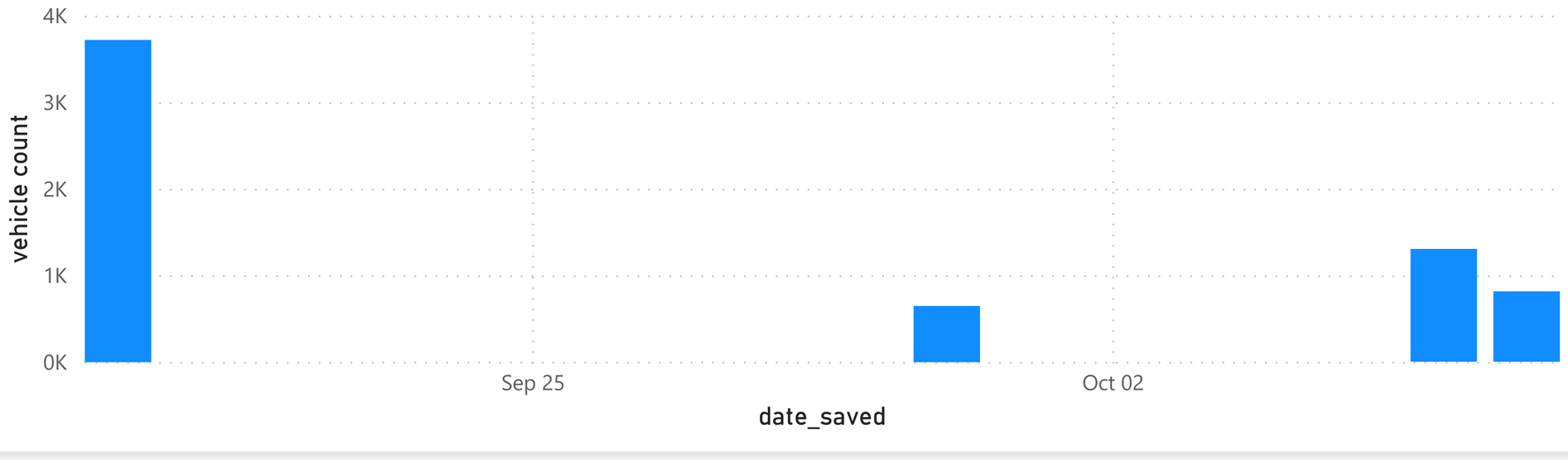


Sensor 1

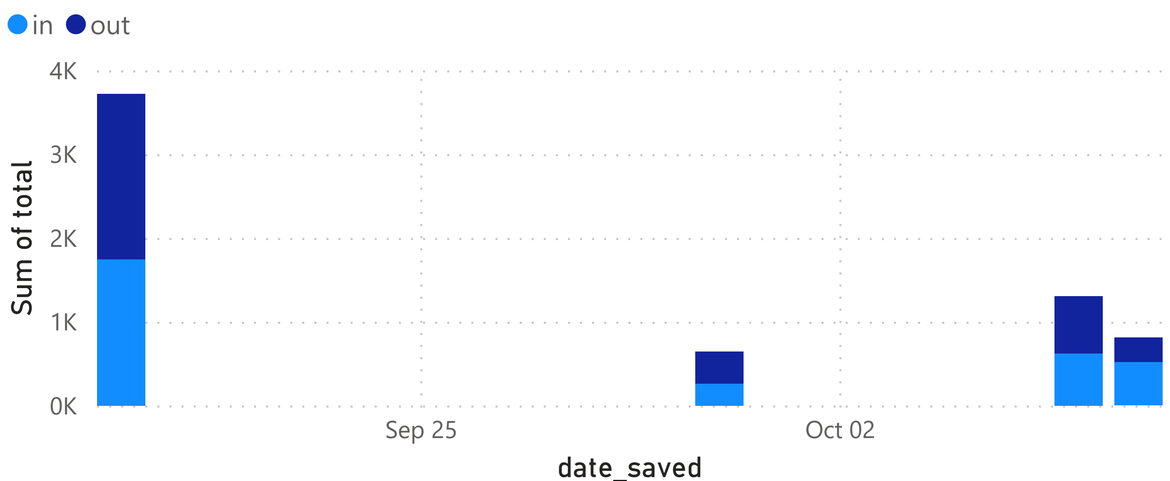
6480

Total vehicles counted

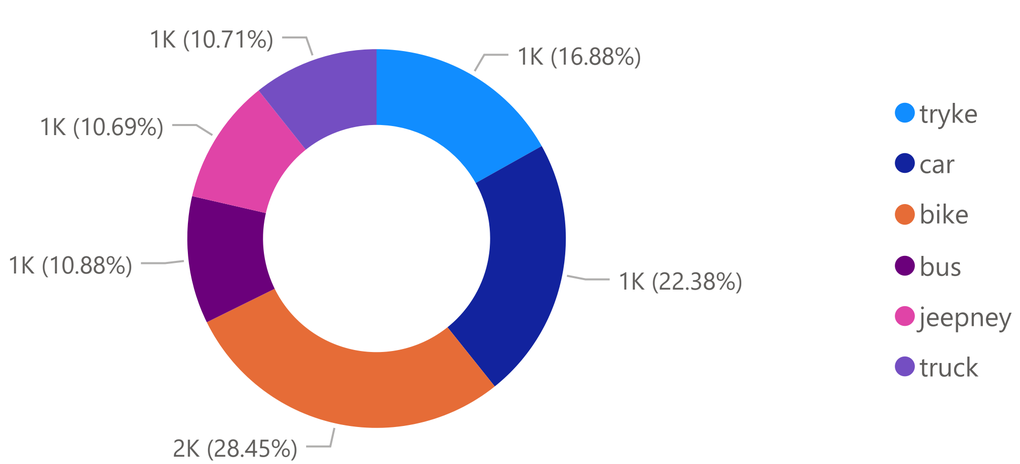
Vehicle count per day



Vehicle actions per day



Vehicle type count for sensor 1

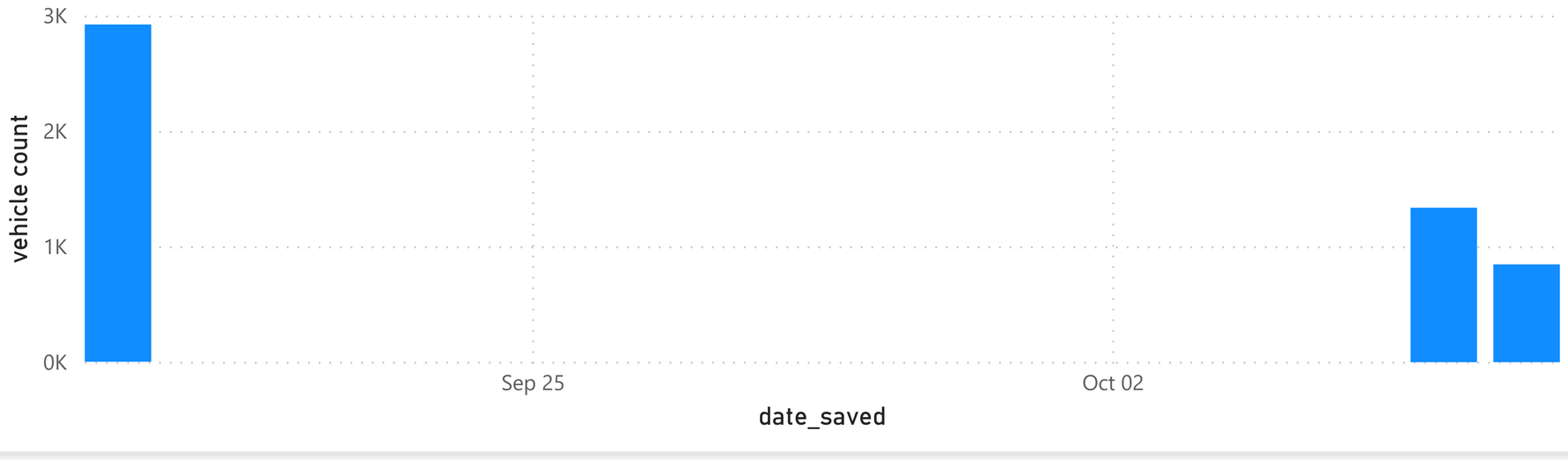


Sensor 2

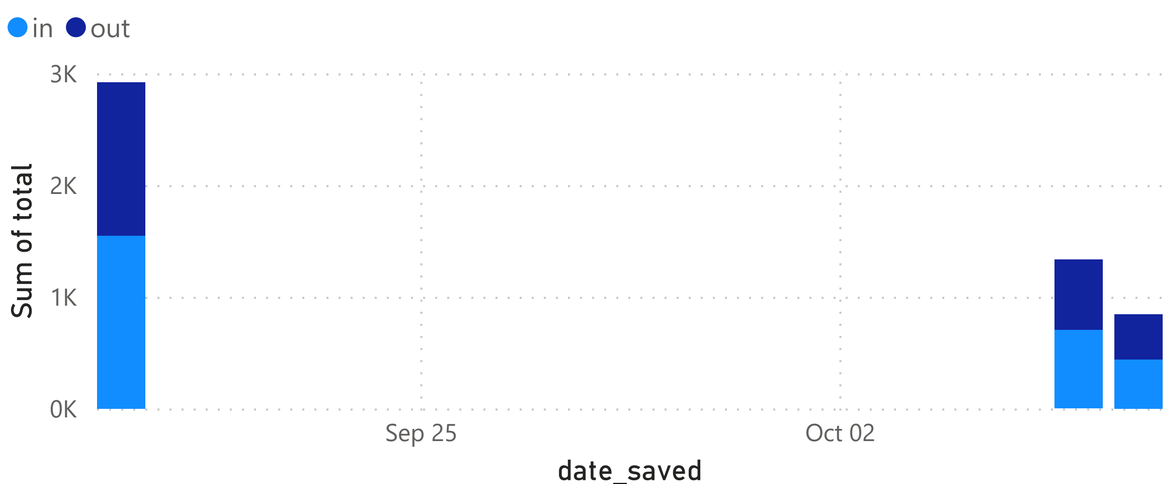
5103

Total vehicles counted

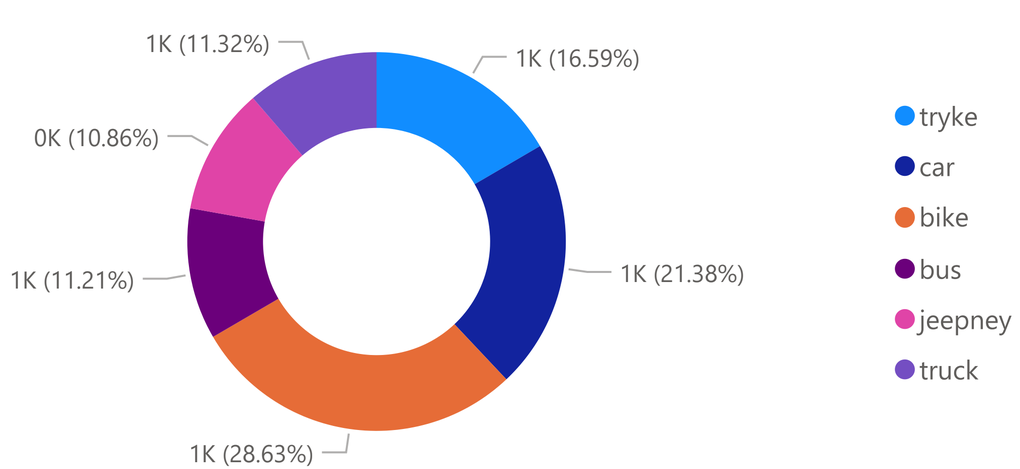
Vehicle count per day



Vehicle actions per day



Vehicle type count for sensor 2

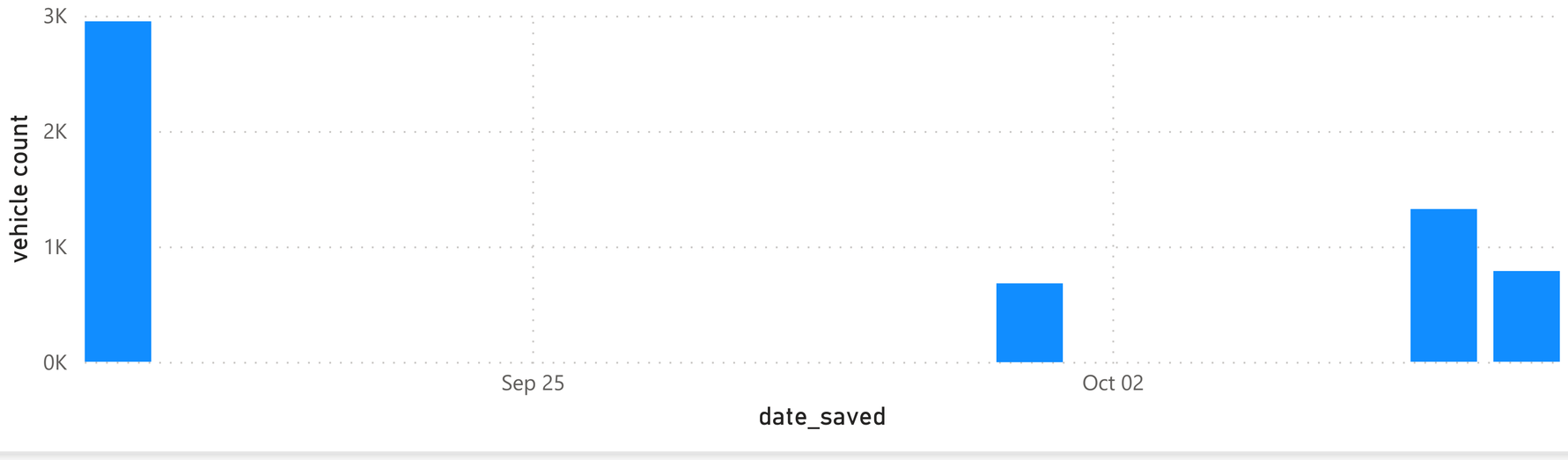


Sensor 3

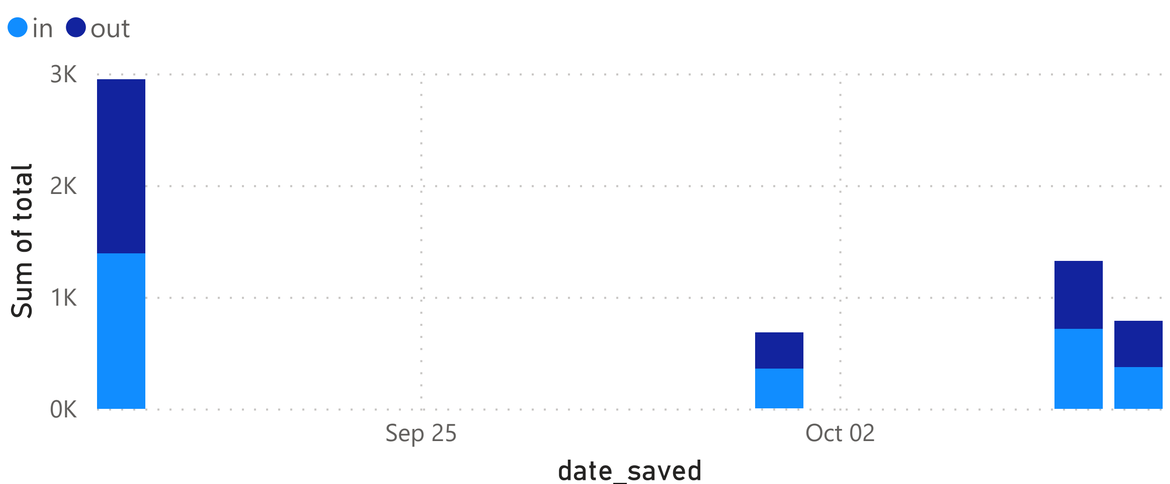
5737

Total vehicles counted

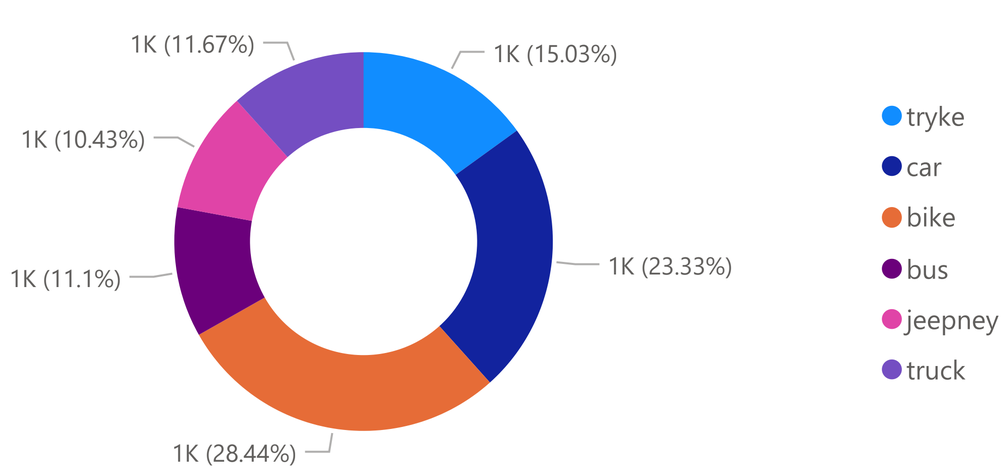
Vehicle count per day



Vehicle actions per day



Vehicle type count for sensor 3

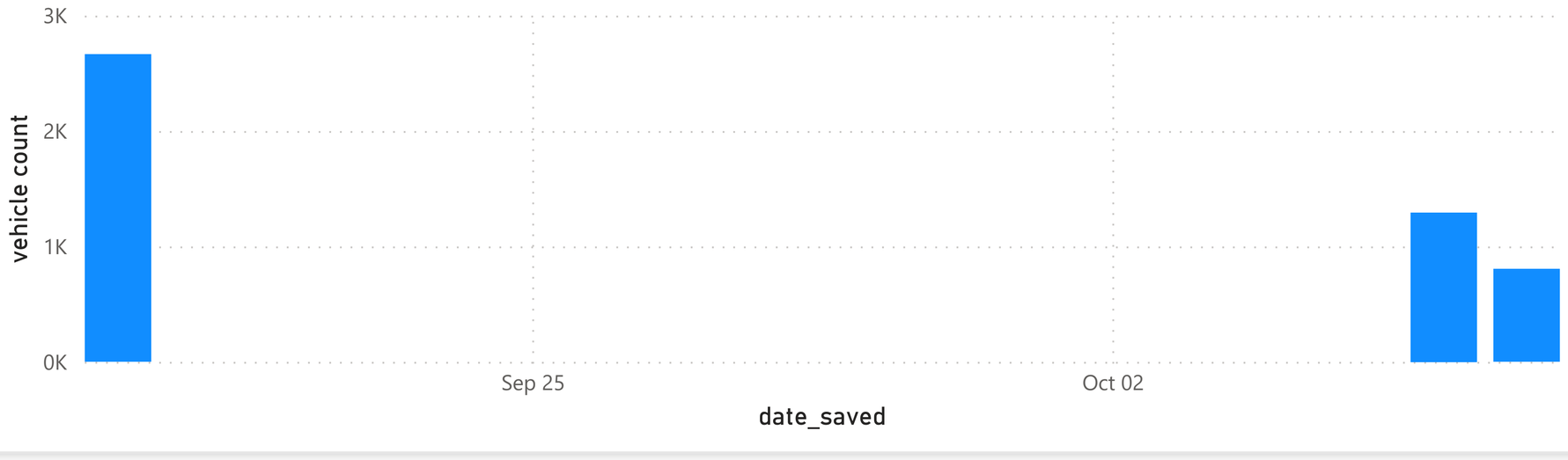


Sensor 4

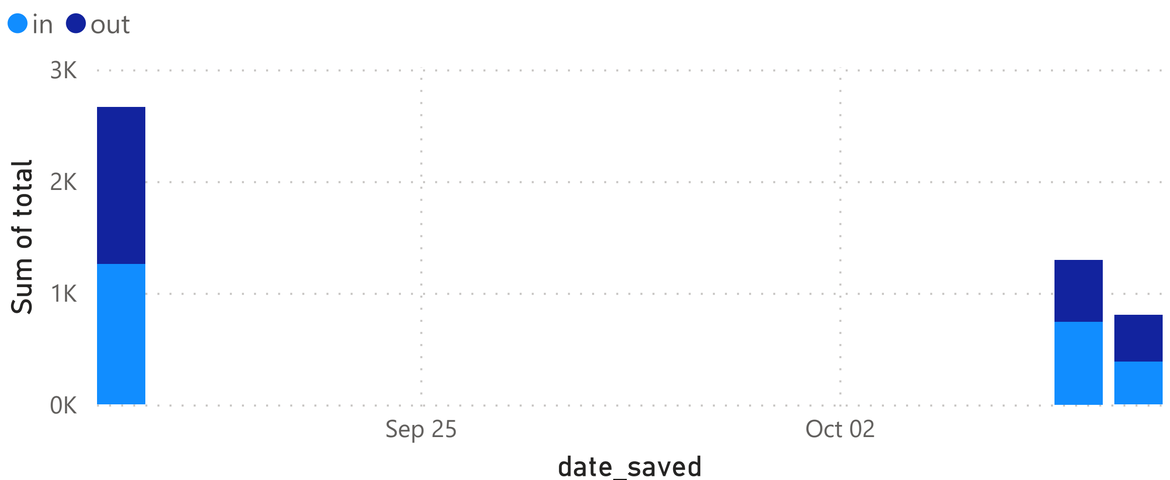
4764

Total vehicles counted

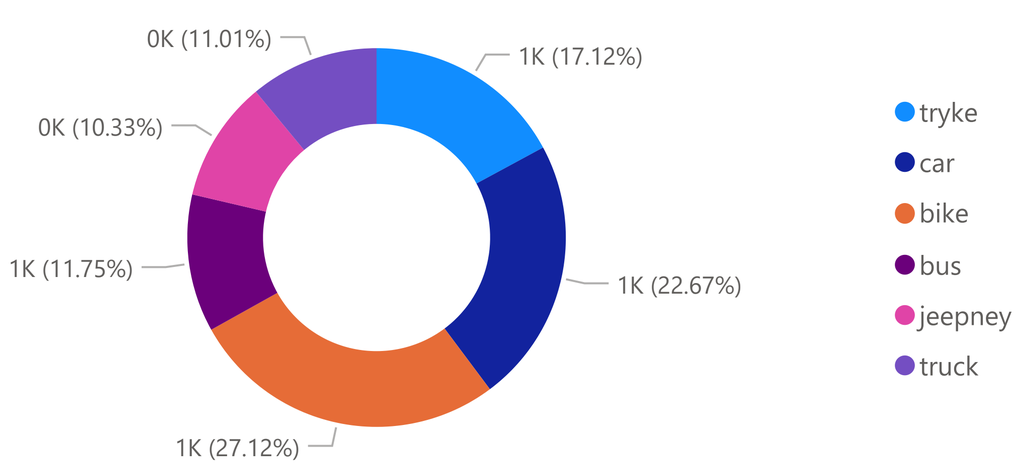
Vehicle count per day



Vehicle actions per day



Vehicle type count for sensor 4

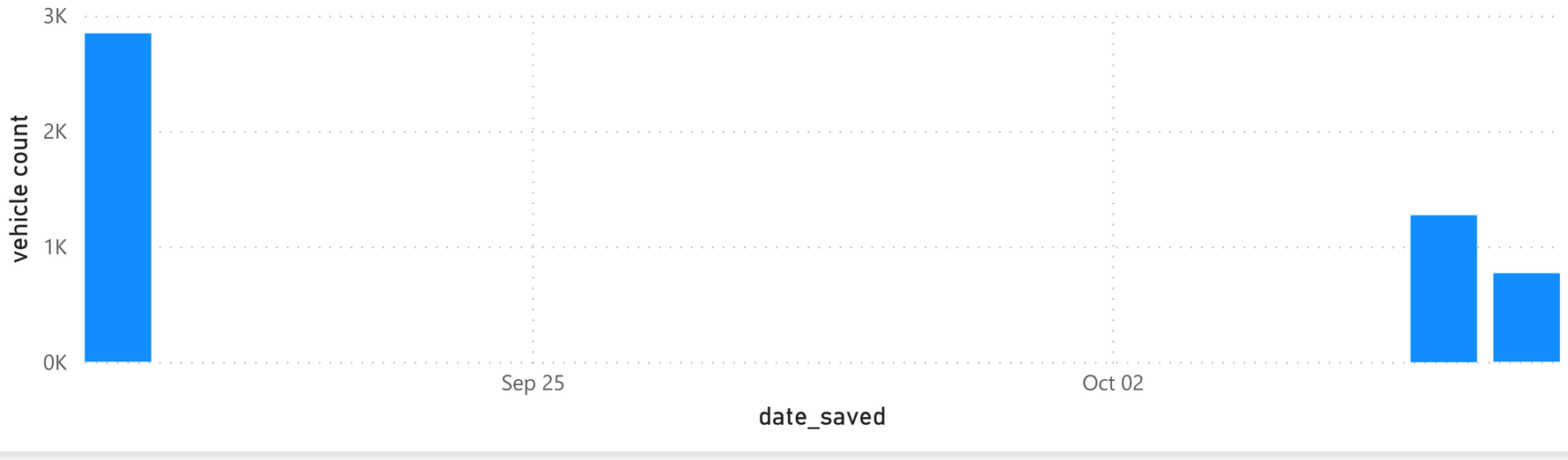


Sensor 5

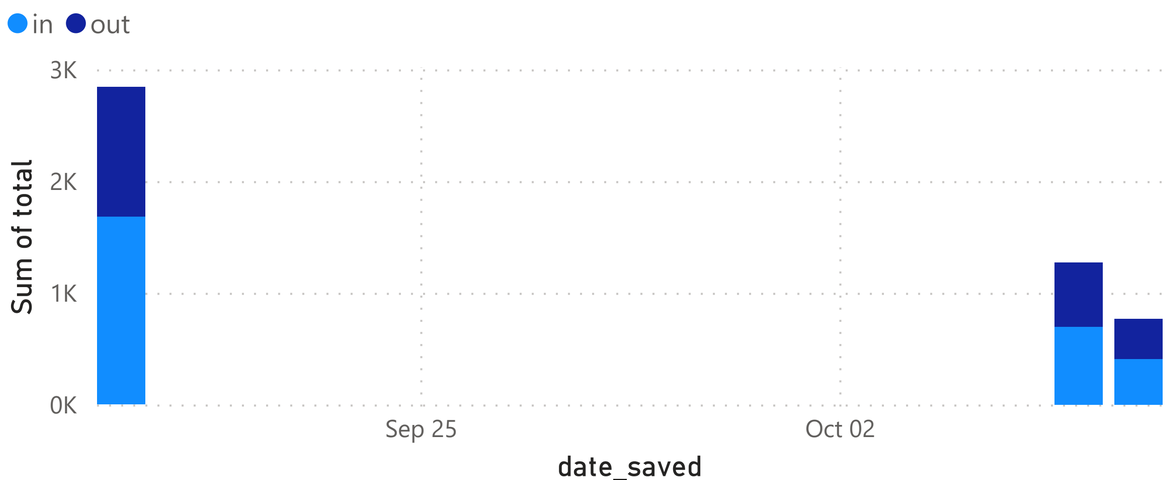
4883

Total vehicles counted

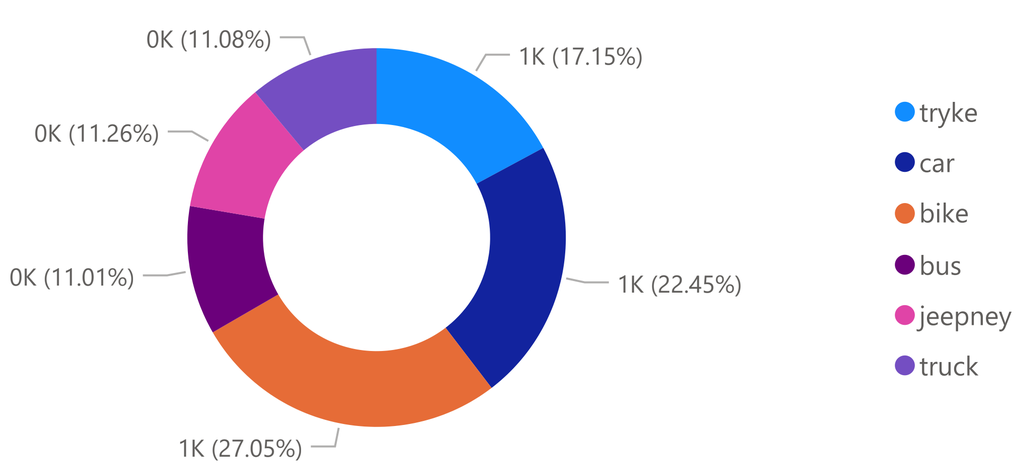
Vehicle count per day



Vehicle actions per day



Vehicle type count for sensor 5

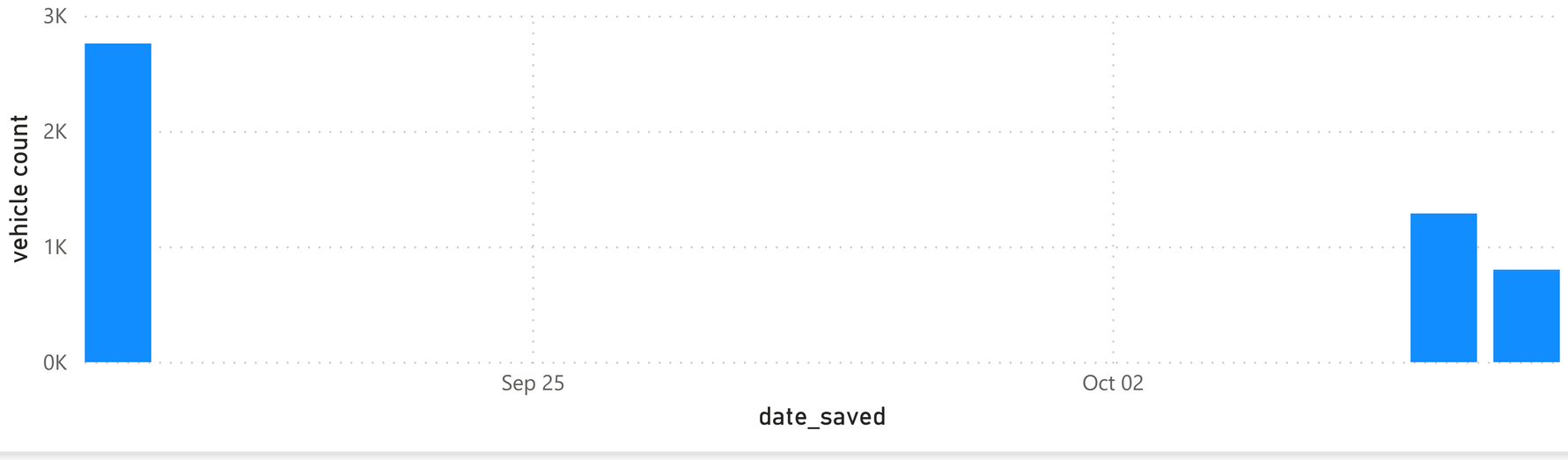


Sensor 6

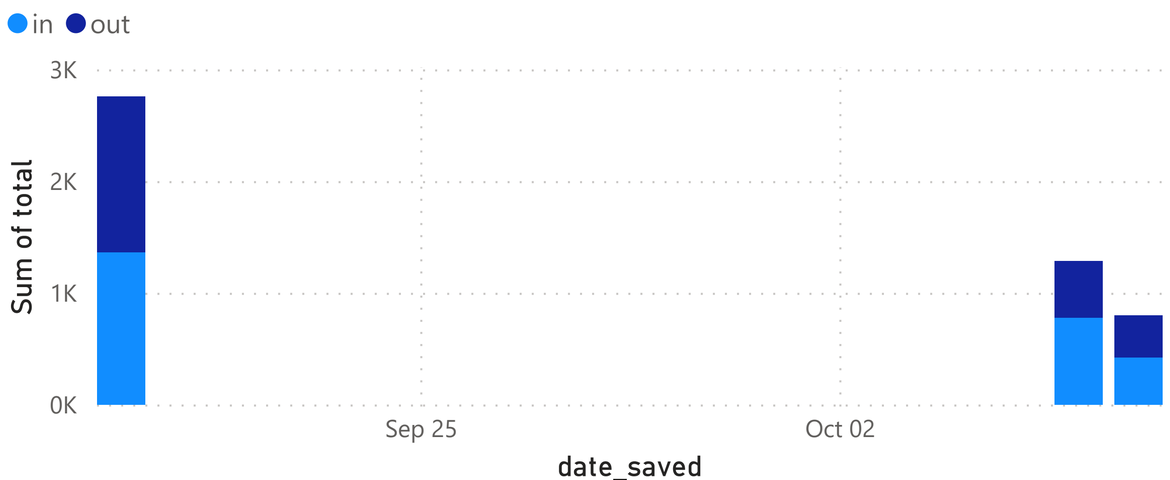
4842

Total vehicles counted

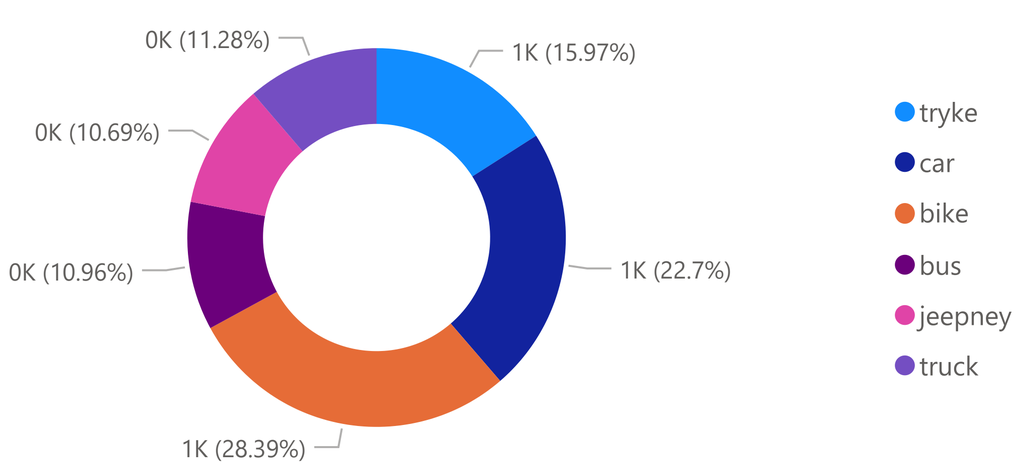
Vehicle count per day



Vehicle actions per day



Vehicle type count for sensor 6



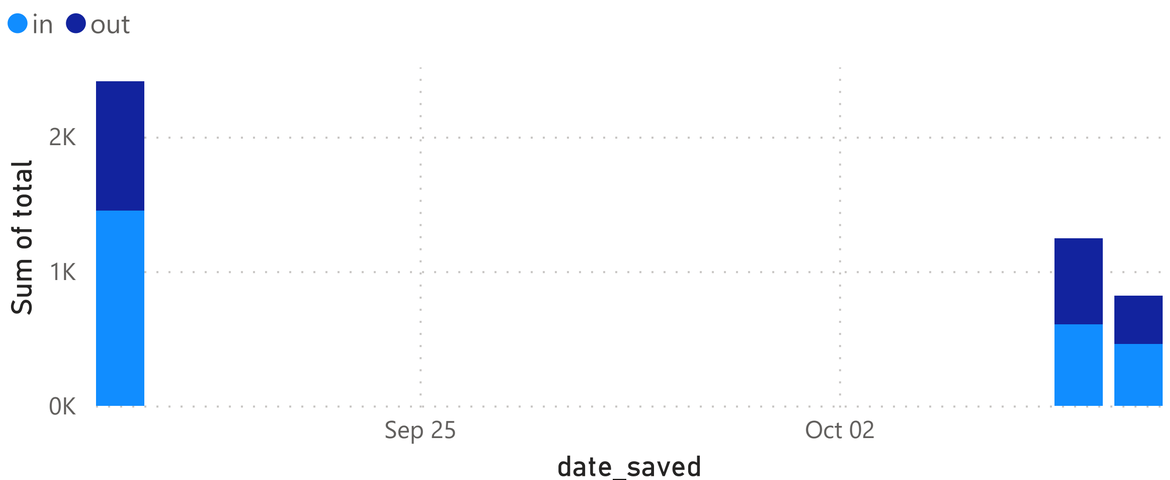
Sensor 7

4468

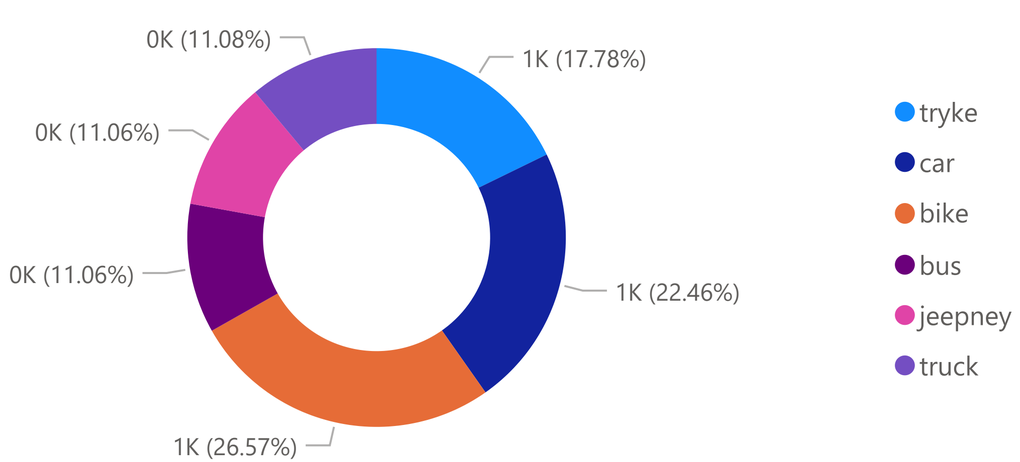
Total vehicles counted

Vehicle count per day

Vehicle actions per day



Vehicle type count for sensor 7

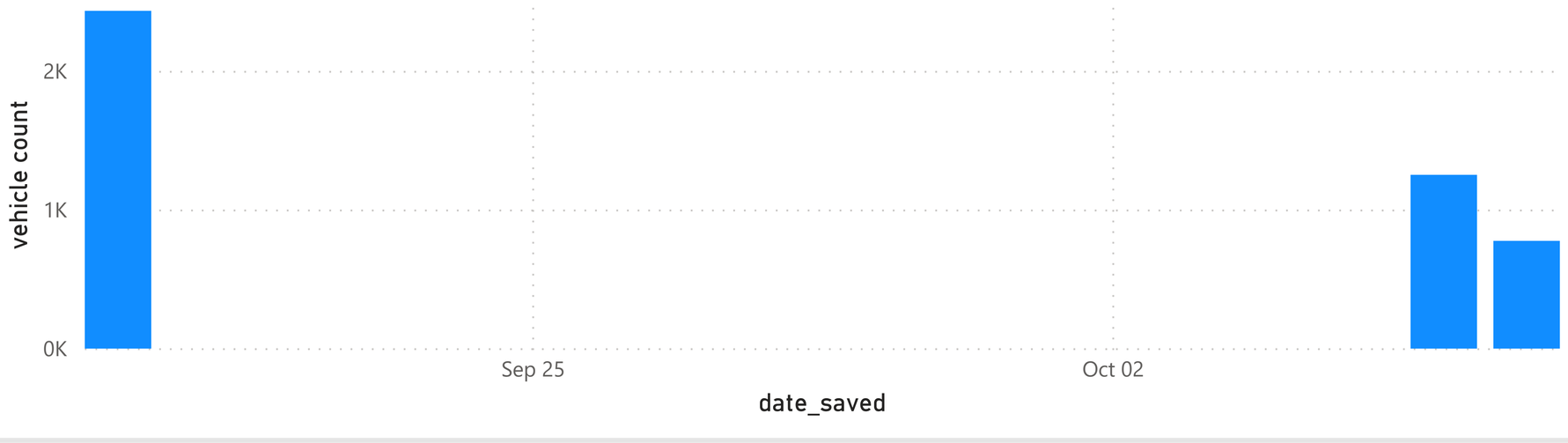


Sensor 8

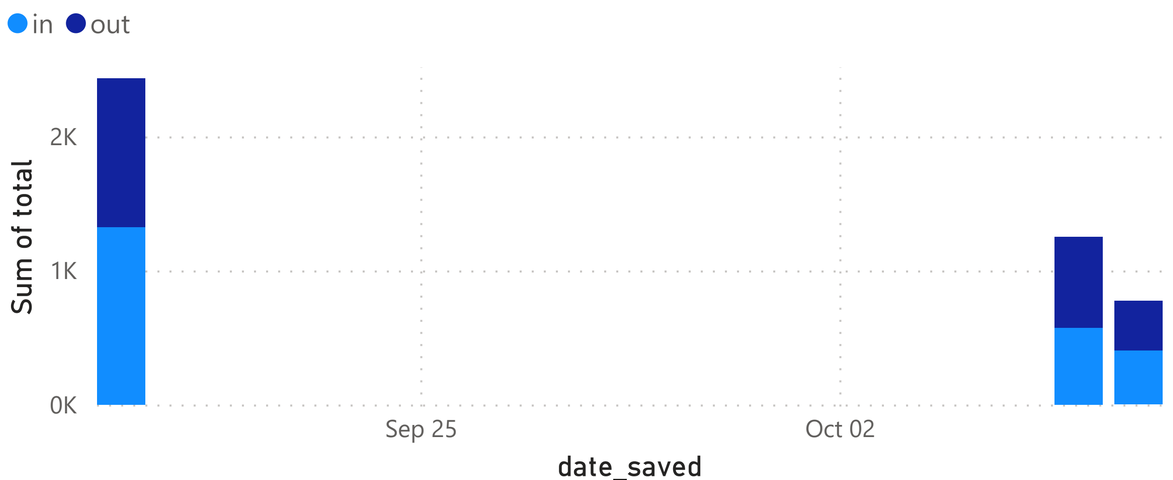
4458

Total vehicles counted

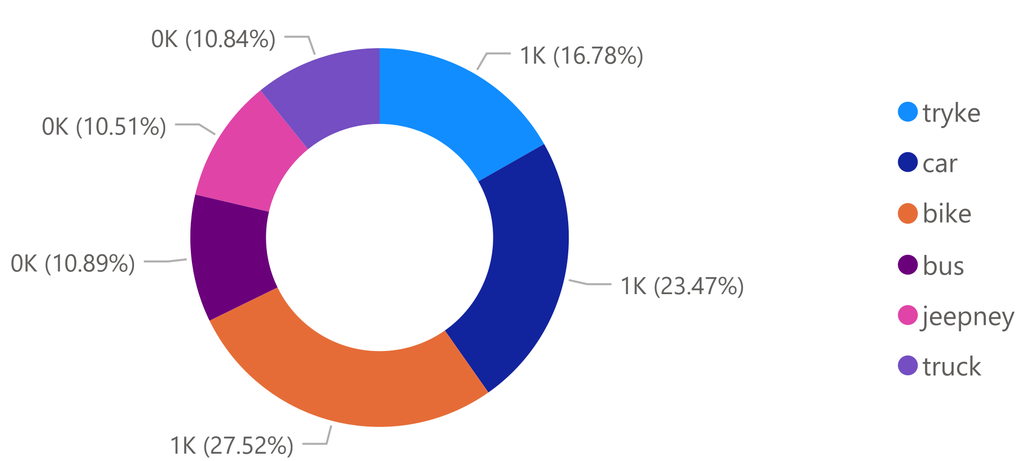
Vehicle count per day



Vehicle actions per day



Vehicle type count for sensor 8

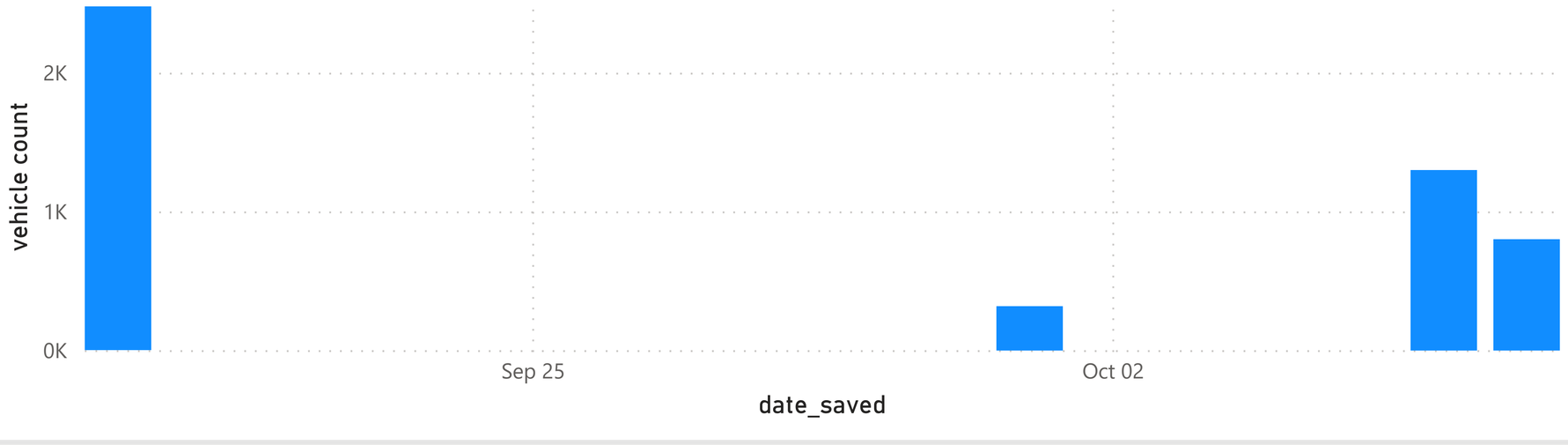


Sensor 9

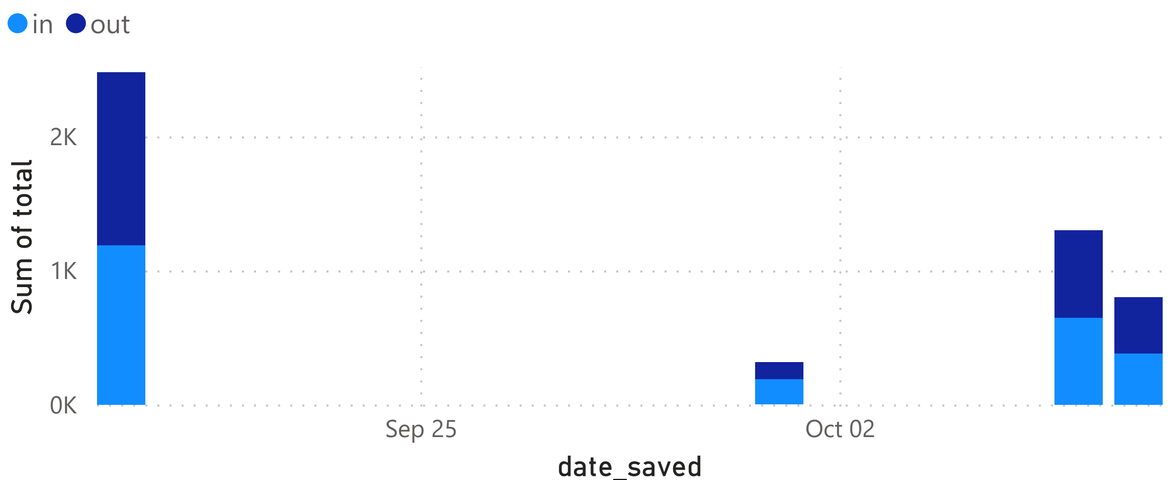
4894

Total vehicles counted

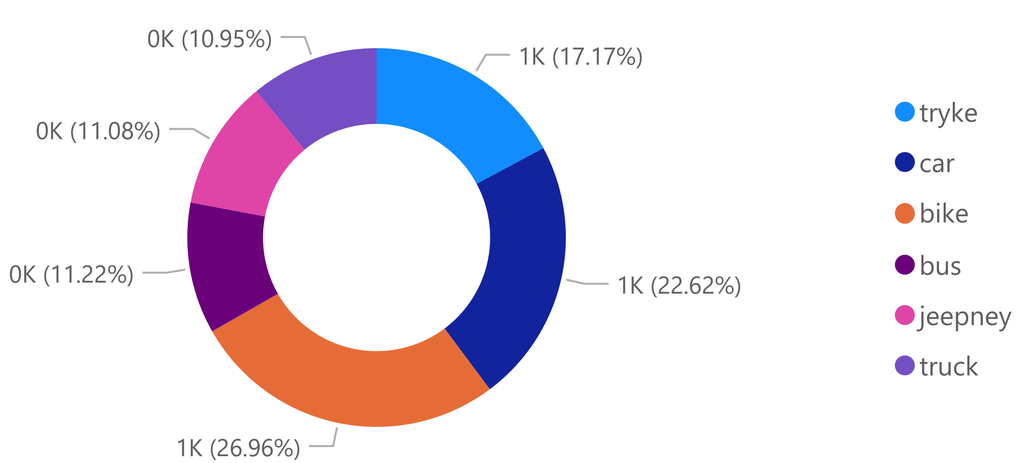
Vehicle count per day



Vehicle actions per day



Vehicle type count for sensor 9

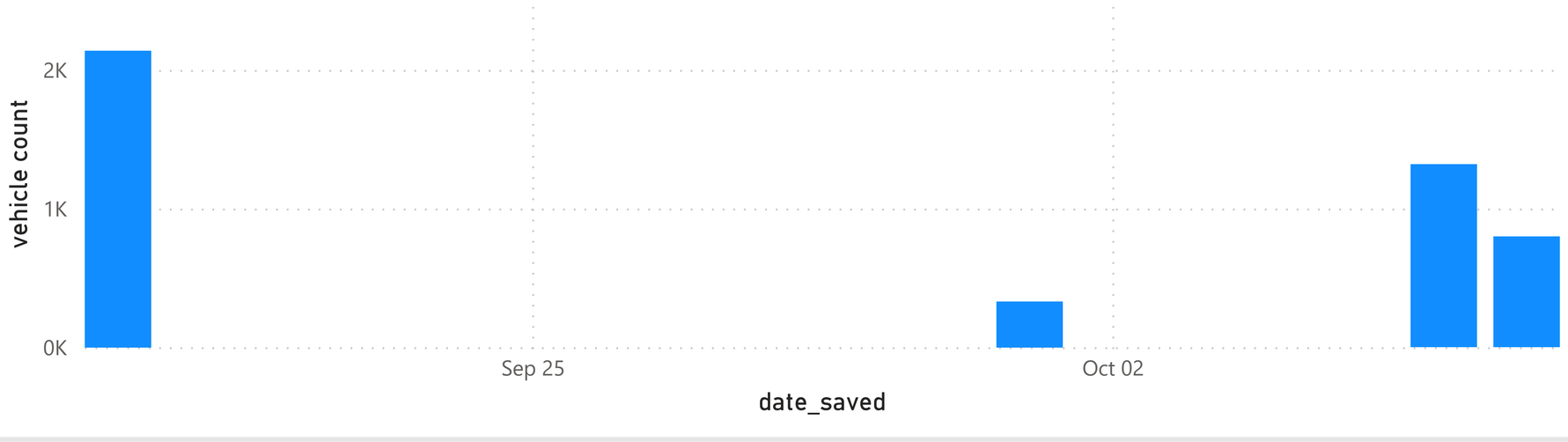


Sensor 10

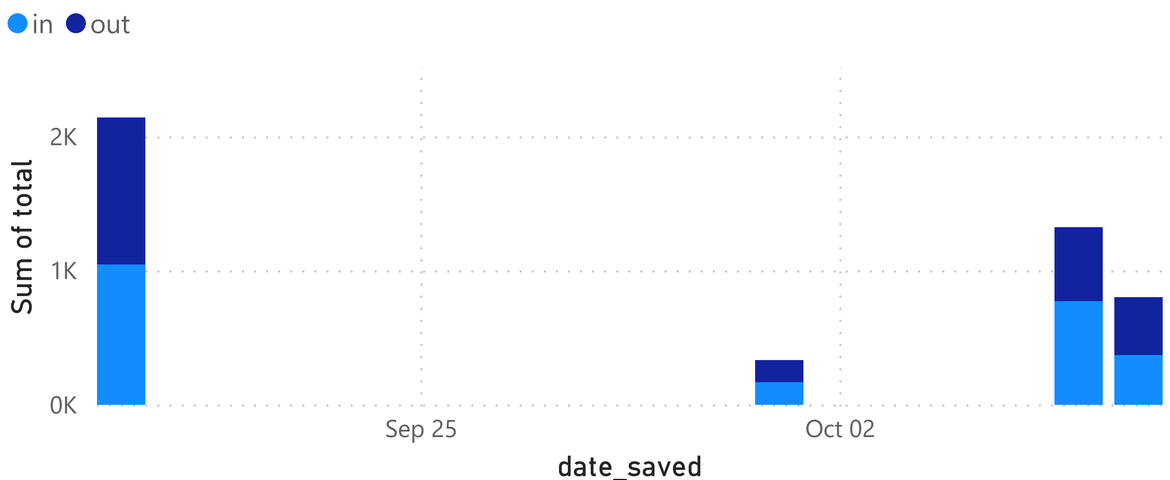
4590

Total vehicles counted

Vehicle count per day



Vehicle actions per day



Vehicle type count for sensor 10