

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

Initialize Kafka

Prepare the consumer and producer programs

[Snippet of the Consumer script](#)

[Snippets of the Producer scripts](#)

Running producers Simultaneously

Initialize the Cassandra Environment

Initialize the Spark Environment

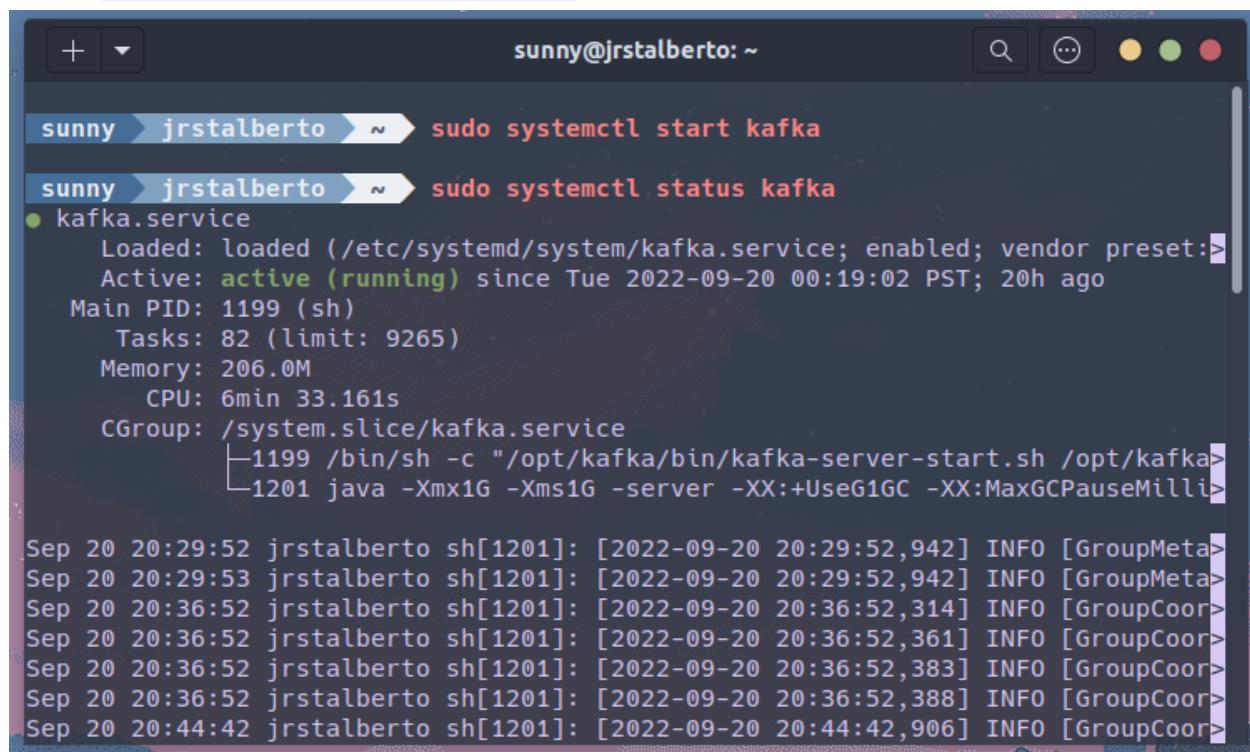
Initialize Kafka

Start kafka from the terminal using

```
sudo systemctl start kafka
```

And check if kafka started successfully

```
sudo systemctl status kafka
```



The screenshot shows a terminal window titled 'sunny@jrstalberto: ~'. The terminal displays the following command and its output:

```
sunny ~ % sudo systemctl start kafka
[sudo] password for sunny:
sunny ~ % sudo systemctl status kafka
● kafka.service
  Loaded: loaded (/etc/systemd/system/kafka.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2022-09-20 00:19:02 PST; 20h ago
    Main PID: 1199 (sh)
      Tasks: 82 (limit: 9265)
     Memory: 206.0M
        CPU: 6min 33.161s
       CGroup: /system.slice/kafka.service
               └─1199 /bin/sh -c "/opt/kafka/bin/kafka-server-start.sh /opt/kafka...
                 ├─1201 java -Xmx1G -Xms1G -server -XX:+UseG1GC -XX:MaxGCPauseMilli...
...
Sep 20 20:29:52 jrstalberto sh[1201]: [2022-09-20 20:29:52,942] INFO [GroupMeta...
Sep 20 20:29:53 jrstalberto sh[1201]: [2022-09-20 20:29:52,942] INFO [GroupMeta...
Sep 20 20:36:52 jrstalberto sh[1201]: [2022-09-20 20:36:52,314] INFO [GroupCoor...
Sep 20 20:36:52 jrstalberto sh[1201]: [2022-09-20 20:36:52,361] INFO [GroupCoor...
Sep 20 20:36:52 jrstalberto sh[1201]: [2022-09-20 20:36:52,383] INFO [GroupCoor...
Sep 20 20:36:52 jrstalberto sh[1201]: [2022-09-20 20:36:52,388] INFO [GroupCoor...
Sep 20 20:44:42 jrstalberto sh[1201]: [2022-09-20 20:44:42,906] INFO [GroupCoor...
```

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
```

```
sunny@jrstalberto: ~
[sudo] password for sunny:

sunny@jrstalberto: ~ sudo systemctl start kafka
[sudo] password for sunny:

sunny@jrstalberto: ~ sudo systemctl status kafka
● kafka.service
    Loaded: loaded (/etc/systemd/system/kafka.service; enabled; vendor preset: true)
    Active: active (running) since Tue 2022-09-20 00:19:02 PST; 20h ago
      Main PID: 1199 (sh)
        Tasks: 82 (limit: 9265)
       Memory: 206.1M
          CPU: 6min 34.353s
        CGroup: /system.slice/kafka.service
                  └─1199 /bin/sh -c "/opt/kafka/bin/kafka-server-start.sh /opt/kafka>
                     └─1201 java -Xmx1G -Xms1G -server -XX:+UseG1GC -XX:MaxGCPauseMilli>

Sep 20 20:29:52 jrstalberto sh[1201]: [2022-09-20 20:29:52,942] INFO [GroupMeta>
Sep 20 20:29:53 jrstalberto sh[1201]: [2022-09-20 20:29:52,942] INFO [GroupMeta>
Sep 20 20:36:52 jrstalberto sh[1201]: [2022-09-20 20:36:52,314] INFO [GroupCoor>
Sep 20 20:36:52 jrstalberto sh[1201]: [2022-09-20 20:36:52,361] INFO [GroupCoor>
Sep 20 20:36:52 jrstalberto sh[1201]: [2022-09-20 20:36:52,383] INFO [GroupCoor>
Sep 20 20:36:52 jrstalberto sh[1201]: [2022-09-20 20:36:52,388] INFO [GroupCoor>
```

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
```

```
sunny@jrstalberto: ~
```

Verify if the topic was created successfully by listing the topics

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

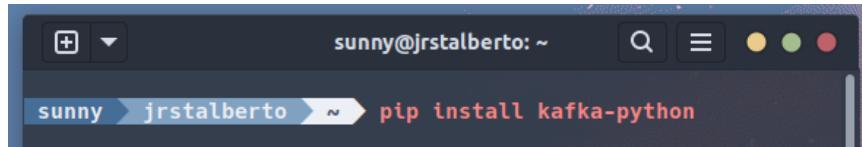
```
sunny@jrstalberto: ~
sunny > jrstalberto ~ kafka-topics.sh --list --bootstrap-server
localhost:9092
__consumer_offsets
cctv_vehicle_counts
sample

sunny@jrstalberto: ~
```

Prepare the consumer and producer programs

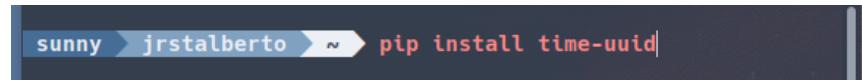
Install the required packages for the downloaded scripts

```
pip install kafka-python
```



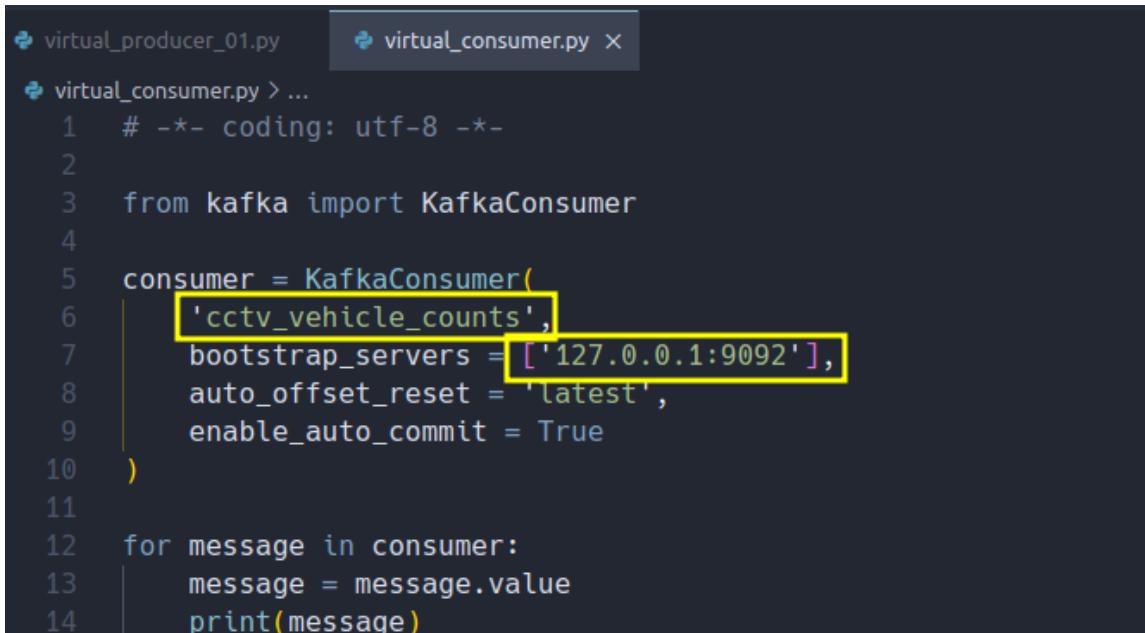
A screenshot of a terminal window titled "sunny@jrstalberto: ~". The command "pip install kafka-python" is being typed into the terminal.

```
pip install time-uuid
```



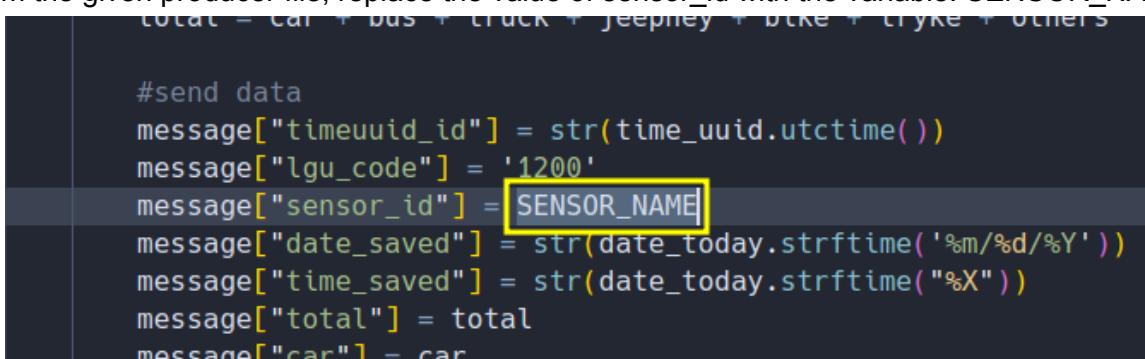
A screenshot of a terminal window titled "sunny@jrstalberto: ~". The command "pip install time-uuid" is being typed into the terminal.

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



```
virtual_producer_01.py virtual_consumer.py x
virtual_consumer.py > ...
1 # -*- coding: utf-8 -*-
2
3 from kafka import KafkaConsumer
4
5 consumer = KafkaConsumer(
6     'cctv_vehicle_counts',
7     bootstrap_servers = ['127.0.0.1:9092'],
8     auto_offset_reset = 'latest',
9     enable_auto_commit = True
10 )
11
12 for message in consumer:
13     message = message.value
14     print(message)
```

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



```
total = car + bus + truck + jeepney + bike + trike + others

#send data
message["timeuuid_id"] = str(time_uuid.utcnow())
message["lgu_code"] = '1200'
message["sensor_id"] = SENSOR_NAME
message["date_saved"] = str(date_today.strftime(' %m/%d/%Y '))
message["time_saved"] = str(date_today.strftime("%X"))
message["total"] = total
message["car"] = car
```

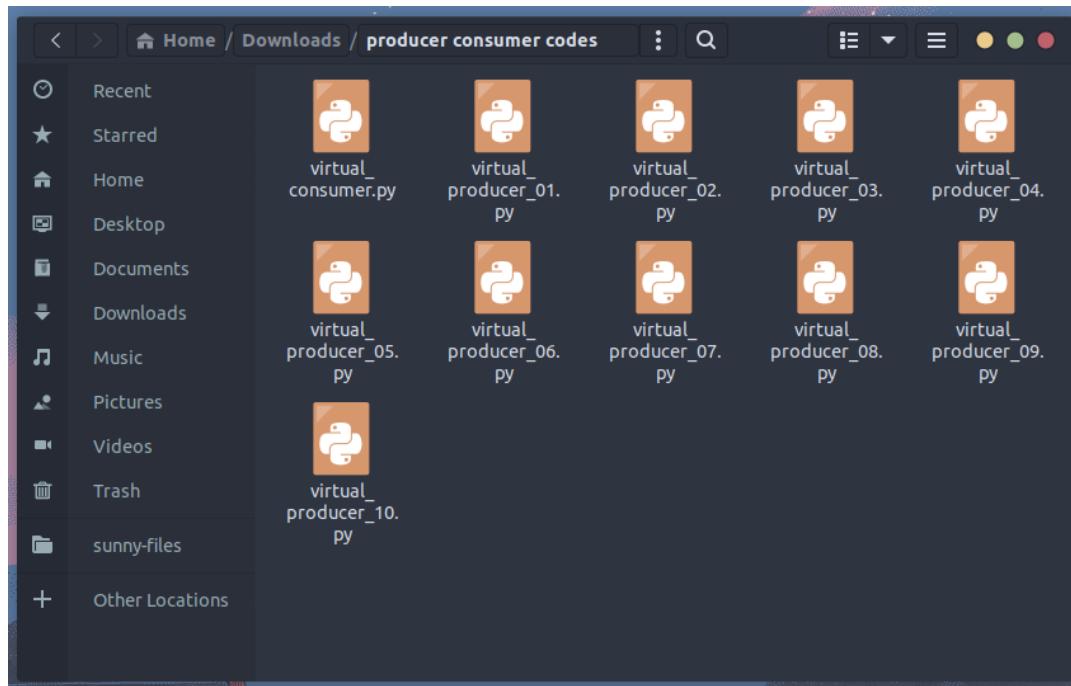
Create the variable SENSOR_NAME and supply the appropriate value.

Verify that the topic name and server are correct.

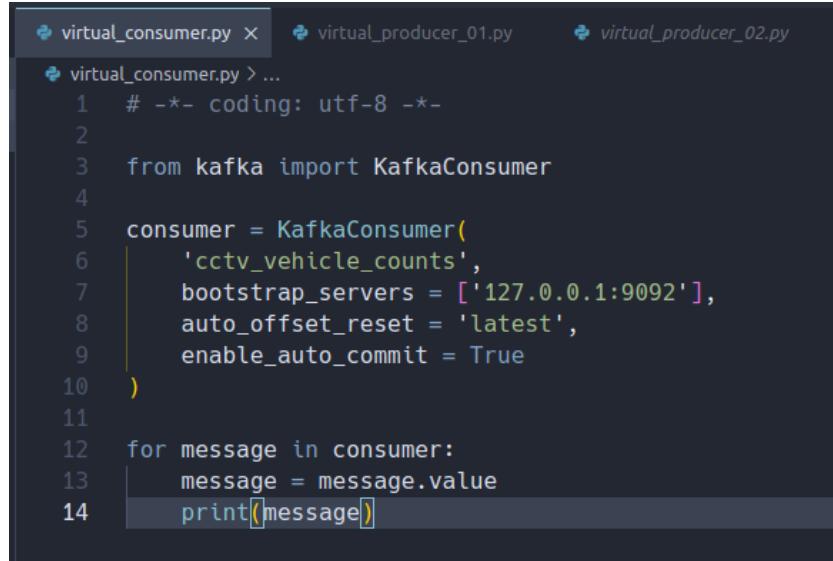
```
1 from json import dumps
2
3 import random
4
5 import time
6
7 import time_uuid, uuid
8
9 KAFKA_TOPIC_NAME_CONS = ["cctv vehicle counts"] #topic
10 KAFKA_BOOTSTRAP_SERVERS_CONS = ['127.0.0.1:9092']
11 SENSOR_NAME = "sensor_01"
12
13 if __name__ == "__main__":
14     print("Kafka Producer Application Started ... ")
15     kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTSTRAP_SERVERS_CONS)
16
17     message_list = [
```

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

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

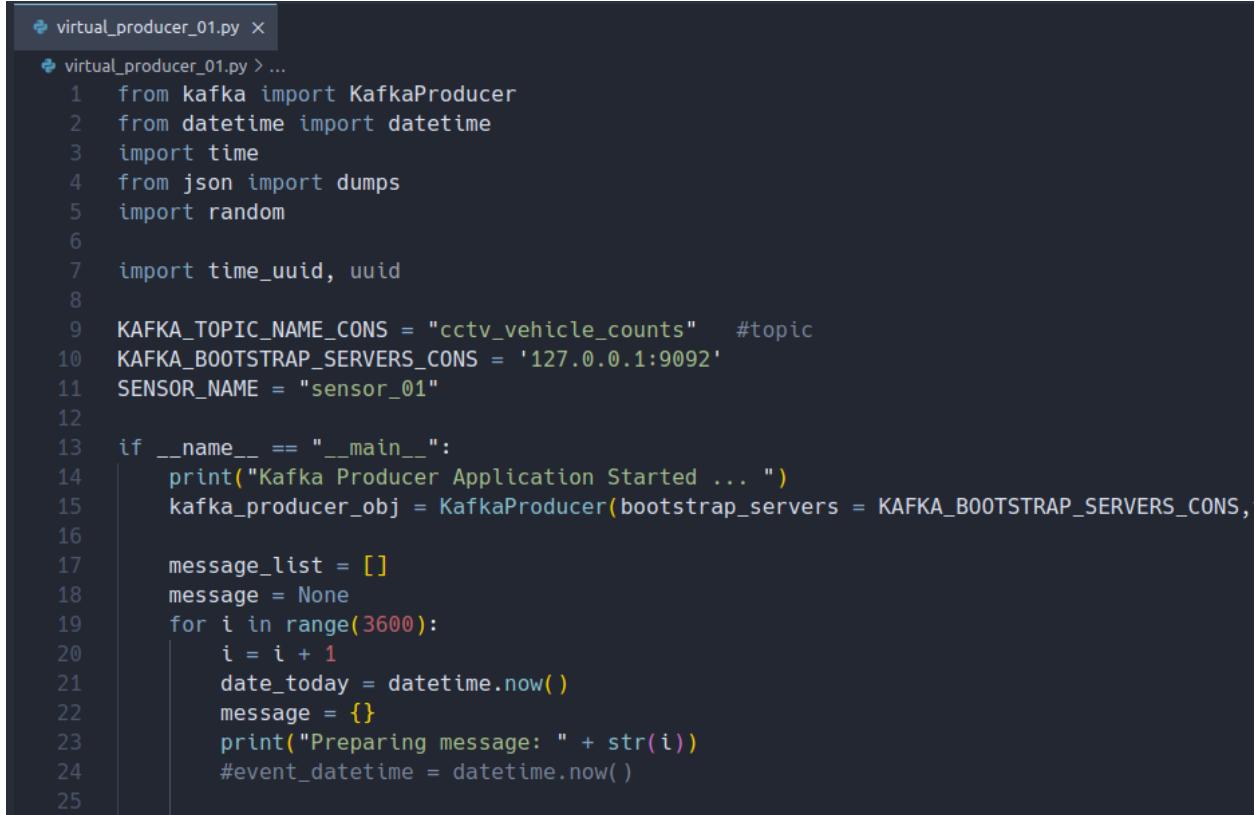


A screenshot of a code editor showing a Python script named `virtual_consumer.py`. The script imports `KafkaConsumer` from `kafka` and defines a consumer object with the topic `'cctv_vehicle_counts'`, bootstrap servers `['127.0.0.1:9092']`, auto offset reset `'latest'`, and enable auto commit `True`. It then loops through messages, prints the value, and highlights the `print(message)` line.

```
# -*- coding: utf-8 -*-
from kafka import KafkaConsumer
consumer = KafkaConsumer(
    'cctv_vehicle_counts',
    bootstrap_servers = [ '127.0.0.1:9092' ],
    auto_offset_reset = 'latest',
    enable_auto_commit = True
)
for message in consumer:
    message = message.value
    print(message)
```

Snippets of the Producer scripts

Producer 1



A screenshot of a code editor showing a Python script named `virtual_producer_01.py`. The script imports KafkaProducer, datetime, time, json, random, and time_uuid. It defines constants for the topic `"cctv_vehicle_counts"`, bootstrap servers `'127.0.0.1:9092'`, and sensor name `"sensor_01"`. It then initializes a KafkaProducer object and enters a loop where it prepares 3600 messages, each containing a timestamp and a value, and prints the preparation message.

```
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_01"

if __name__ == "__main__":
    print("Kafka Producer Application Started ... ")
    kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTSTRAP_SERVERS_CONS,
    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 2

```
virtual_producer_02.py > ...
virtual_producer_02.py > ...
1  from kafka import KafkaProducer
2  from datetime import datetime
3  import time
4  from json import dumps
5  import random
6
7  import time_uuid, uuid
8
9  KAFKA_TOPIC_NAME_CONS = "cctv_vehicle_counts"    #topic
10 KAFKA_BOOTSTRAP_SERVERS_CONS = '127.0.0.1:9092'
11 SENSOR_NAME = "sensor_02"
12
13 if __name__ == "__main__":
14     print("Kafka Producer Application Started ... ")
15     kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTSTRAP_SERVERS_CONS,value_serializer=lambda x: dumps(x).encode('utf-8'))
16
17     message_list = []
18     message = None
19     for i in range(3600):
20         i = i + 1
21         date_today = datetime.now()
22         message = {}
23         print("Preparing message: " + str(i))
24         #event_datetime = datetime.now()
```

Producer 3

```
virtual_producer_03.py > ...
virtual_producer_03.py > ...
1  from kafka import KafkaProducer
2  from datetime import datetime
3  import time
4  from json import dumps
5  import random
6
7  import time_uuid, uuid
8
9  KAFKA_TOPIC_NAME_CONS = "cctv_vehicle_counts"    #topic
10 KAFKA_BOOTSTRAP_SERVERS_CONS = '127.0.0.1:9092'
11 SENSOR_NAME = "sensor_03"
12
13 if __name__ == "__main__":
14     print("Kafka Producer Application Started ... ")
15     kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTSTRAP_SERVERS_CONS,value_serializer=lambda x: dumps(x).encode('utf-8'))
16
17     message_list = []
18     message = None
19     for i in range(3600):
20         i = i + 1
21         date_today = datetime.now()
22         message = {}
23         print("Preparing message: " + str(i))
24         #event_datetime = datetime.now()
```

Producer 4

Producer 5

```
virtual_producer_05.py x
virtual_producer_05.py > ...
1  from kafka import KafkaProducer
2  from datetime import datetime
3  import time
4  from json import dumps
5  import random
6
7  import time_uuid, uuid
8
9  KAFKA_TOPIC_NAME_CONS = "cctv_vehicle_counts"    #topic
10 KAFKA_BOOTSTRAP_SERVERS_CONS = '127.0.0.1:9092'
11 SENSOR_NAME = "sensor_05"
12
13 if __name__ == "__main__":
14     print("Kafka Producer Application Started ... ")
15     kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTSTRAP_SERVERS_CONS,
16                                         value_serializer=lambda x: dumps(x).encode('utf-8'))
17
18     message_list = []
19     message = None
20     for i in range(3600):
21         i = i + 1
22         date_today = datetime.now()
23         message = {}
24         print("Preparing message: " + str(i))
25         #event_datetime = datetime.now()
```

Producer 6

```
virtual_producer_06.py > ...
1  from kafka import KafkaProducer
2  from datetime import datetime
3  import time
4  from json import dumps
5  import random
6
7  import time_uuid, uuid
8
9  KAFKA_TOPIC_NAME_CONS = "cctv_vehicle_counts"    #topic
10 KAFKA_BOOTSTRAP_SERVERS_CONS = '127.0.0.1:9092'
11 SENSOR_NAME = "sensor_06"
12
13 if __name__ == "__main__":
14     print("Kafka Producer Application Started ... ")
15     kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTSTRAP_SERVERS_CONS,
16                                         value_serializer=lambda x: dumps(x).encode('utf-8')
17
18     message_list = []
19     message = None
20     for i in range(3600):
21         i = i + 1
22         date_today = datetime.now()
23         message = {}
24         print("Preparing message: " + str(i))
```

Producer 7

```
virtual_producer_07.py > ...
1  from kafka import KafkaProducer
2  from datetime import datetime
3  import time
4  from json import dumps
5  import random
6
7  import time_uuid, uuid
8
9  KAFKA_TOPIC_NAME_CONS = "cctv_vehicle_counts"    #topic
10 KAFKA_BOOTSTRAP_SERVERS_CONS = '127.0.0.1:9092'
11 SENSOR_NAME = "sensor_07"
12
13 if __name__ == "__main__":
14     print("Kafka Producer Application Started ... ")
15     kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTSTRAP_SERVERS_CONS,
16                                         value_serializer=lambda x: dumps(x).encode('utf-8')
17
18     message_list = []
19     message = None
20     for i in range(3600):
21         i = i + 1
22         date_today = datetime.now()
23         message = {}
24         print("Preparing message: " + str(i))
25         #event_datetime = datetime.now()
```

Producer 8

```
virtual_producer_08.py > ...
virtual_producer_08.py > ...
1  from kafka import KafkaProducer
2  from datetime import datetime
3  import time
4  from json import dumps
5  import random
6
7  import time_uuid, uuid
8
9  KAFKA_TOPIC_NAME_CONS = "cctv_vehicle_counts"    #topic
10 KAFKA_BOOTSTRAP_SERVERS_CONS = '127.0.0.1:9092'
11 SENSOR_NAME = "sensor_08"
12
13 if __name__ == "__main__":
14     print("Kafka Producer Application Started ... ")
15     kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTSTRAP_SERVERS_CONS,
16                                         value_serializer=lambda x: dumps(x).encode('utf-8'))
17
18     message_list = []
19     message = None
20     for i in range(3600):
21         i = i + 1
22         date_today = datetime.now()
23         message = {}
24         print("Preparing message: " + str(i))
25         #event_datetime = datetime.now()
```

Producer 9

```
virtual_producer_09.py > ...
virtual_producer_09.py > ...
1  from kafka import KafkaProducer
2  from datetime import datetime
3  import time
4  from json import dumps
5  import random
6
7  import time_uuid, uuid
8
9  KAFKA_TOPIC_NAME_CONS = "cctv_vehicle_counts"    #topic
10 KAFKA_BOOTSTRAP_SERVERS_CONS = '127.0.0.1:9092'
11 SENSOR_NAME = "sensor_09"
12
13 if __name__ == "__main__":
14     print("Kafka Producer Application Started ... ")
15     kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTSTRAP_SERVERS_CONS,
16                                         value_serializer=lambda x: dumps(x).encode('utf-8'))
17
18     message_list = []
19     message = None
20     for i in range(3600):
21         i = i + 1
22         date_today = datetime.now()
23         message = {}
24         print("Preparing message: " + str(i))
25         #event_datetime = datetime.now()
```

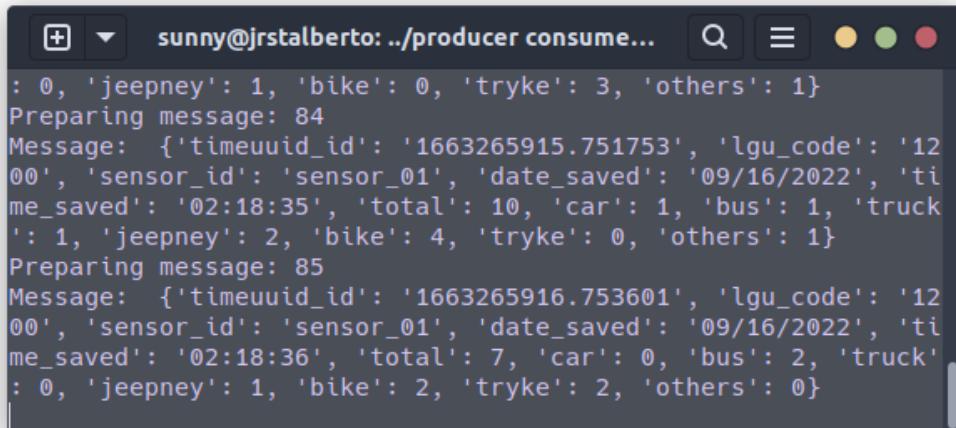
Producer 10

```
virtual_producer_10.py > ...
1  from kafka import KafkaProducer
2  from datetime import datetime
3  import time
4  from json import dumps
5  import random
6
7  import time_uuid, uuid
8
9  KAFKA_TOPIC_NAME_CONS = "cctv_vehicle_counts"    #topic
10 KAFKA_BOOTSTRAP_SERVERS_CONS = '127.0.0.1:9092'
11 SENSOR_NAME = "sensor_10"
12
13 if __name__ == "__main__":
14     print("Kafka Producer Application Started ... ")
15     kafka_producer_obj = KafkaProducer(bootstrap_servers = KAFKA_BOOTSTRAP_SERVERS_CONS,
16                                         value_serializer=lambda x: dumps(x).encode('utf-8'))
17
18     message_list = []
19     message = None
20     for i in range(3600):
21         i = i + 1
22         date_today = datetime.now()
23         message = {}
24         print("Preparing message: " + str(i))
25         #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



A screenshot of a terminal window titled "sunny@jrstalberto: ..producer consume...". The window displays the output of a Python script. It shows two messages being prepared for sending. The first message is for index 84, containing vehicle counts for jeepney, bike, tryke, and others. The second message is for index 85, also containing vehicle counts for the same categories. The terminal interface includes a header with a search bar and three colored dots (yellow, green, red) on the right.

```
: 0, 'jeepney': 1, 'bike': 0, 'tryke': 3, 'others': 1}
Preparing message: 84
Message: {'timeuuid_id': '1663265915.751753', 'lgu_code': '12
00', 'sensor_id': 'sensor_01', 'date_saved': '09/16/2022', 'ti
me_saved': '02:18:35', 'total': 10, 'car': 1, 'bus': 1, 'truck
': 1, 'jeepney': 2, 'bike': 4, 'tryke': 0, 'others': 1}
Preparing message: 85
Message: {'timeuuid_id': '1663265916.753601', 'lgu_code': '12
00', 'sensor_id': 'sensor_01', 'date_saved': '09/16/2022', 'ti
me_saved': '02:18:36', 'total': 7, 'car': 0, 'bus': 2, 'truck'
': 0, 'jeepney': 1, 'bike': 2, 'tryke': 2, 'others': 0}
```

Producer 2

```
+ └ sunny@jrstalberto: ..//producer consume... Q ≡ ○ ○ ○
': 1, 'jeepney': 2, 'bike': 0, 'tryke': 2, 'others': 1}
Preparing message: 164
Message: {'timeuuid_id': '1663266002.303559', 'lgu_code': '12
00', 'sensor_id': 'sensor_02', 'date_saved': '09/16/2022', 'ti
me_saved': '02:20:02', 'total': 11, 'car': 0, 'bus': 1, 'truck
': 2, 'jeepney': 1, 'bike': 5, 'tryke': 2, 'others': 0}
Preparing message: 165
Message: {'timeuuid_id': '1663266003.305662', 'lgu_code': '12
00', 'sensor_id': 'sensor_02', 'date_saved': '09/16/2022', 'ti
me_saved': '02:20:03', 'total': 9, 'car': 2, 'bus': 0, 'truck'
: 2, 'jeepney': 0, 'bike': 2, 'tryke': 1, 'others': 2}
```

Producer 3

```
+ └ sunny@jrstalberto: ..//producer consume... Q ≡ ○ ○ ○
': 2, 'jeepney': 2, 'bike': 2, 'tryke': 3, 'others': 2}
Preparing message: 224
Message: {'timeuuid_id': '1663266067.468243', 'lgu_code': '12
00', 'sensor_id': 'sensor_03', 'date_saved': '09/16/2022', 'ti
me_saved': '02:21:07', 'total': 12, 'car': 3, 'bus': 2, 'truck
': 2, 'jeepney': 2, 'bike': 2, 'tryke': 1, 'others': 0}
Preparing message: 225
Message: {'timeuuid_id': '1663266068.469445', 'lgu_code': '12
00', 'sensor_id': 'sensor_03', 'date_saved': '09/16/2022', 'ti
me_saved': '02:21:08', 'total': 8, 'car': 4, 'bus': 0, 'truck'
: 1, 'jeepney': 1, 'bike': 2, 'tryke': 0, 'others': 0}
```

Producer 4

```
+ └ sunny@jrstalberto: ..//producer consume... Q ≡ ○ ○ ○
': 0, 'jeepney': 0, 'bike': 2, 'tryke': 3, 'others': 1}
Preparing message: 248
Message: {'timeuuid_id': '1663266098.236731', 'lgu_code': '12
00', 'sensor_id': 'sensor_04', 'date_saved': '09/16/2022', 'ti
me_saved': '02:21:38', 'total': 7, 'car': 1, 'bus': 0, 'truck
': 0, 'jeepney': 0, 'bike': 2, 'tryke': 2, 'others': 2}
Preparing message: 249
Message: {'timeuuid_id': '1663266099.238211', 'lgu_code': '12
00', 'sensor_id': 'sensor_04', 'date_saved': '09/16/2022', 'ti
me_saved': '02:21:39', 'total': 17, 'car': 3, 'bus': 0, 'truck
': 2, 'jeepney': 2, 'bike': 5, 'tryke': 3, 'others': 2}
```

Producer 5

```
+ └ sunny@jrstalberto: ..//producer consume... Q Ⓜ ⓘ ⓘ
': 1, 'jeepney': 1, 'bike': 1, 'tryke': 1, 'others': 1}
Preparing message: 245
Message: {'timeuuid_id': '1663266111.75419', 'lgu_code': '120
0', 'sensor_id': 'sensor_05', 'date_saved': '09/16/2022', 'tim
e_saved': '02:21:51', 'total': 7, 'car': 0, 'bus': 0, 'truck':
1, 'jeepney': 2, 'bike': 2, 'tryke': 1, 'others': 1}
Preparing message: 246
Message: {'timeuuid_id': '1663266112.755915', 'lgu_code': '12
00', 'sensor_id': 'sensor_05', 'date_saved': '09/16/2022', 'ti
me_saved': '02:21:52', 'total': 11, 'car': 0, 'bus': 0, 'truck
': 0, 'jeepney': 1, 'bike': 5, 'tryke': 3, 'others': 2}
```

Producer 6

```
+ └ sunny@jrstalberto: ..//producer consume... Q Ⓜ ⓘ ⓘ
': 2, 'jeepney': 1, 'bike': 5, 'tryke': 2, 'others': 1}
Preparing message: 252
Message: {'timeuuid_id': '1663266123.457162', 'lgu_code': '12
00', 'sensor_id': 'sensor_06', 'date_saved': '09/16/2022', 'ti
me_saved': '02:22:03', 'total': 14, 'car': 4, 'bus': 0, 'truck
': 1, 'jeepney': 0, 'bike': 4, 'tryke': 3, 'others': 2}
Preparing message: 253
Message: {'timeuuid_id': '1663266124.458233', 'lgu_code': '12
00', 'sensor_id': 'sensor_06', 'date_saved': '09/16/2022', 'ti
me_saved': '02:22:04', 'total': 8, 'car': 4, 'bus': 0, 'truck'
: 1, 'jeepney': 0, 'bike': 1, 'tryke': 1, 'others': 1}
```

Producer 7

```
+ └ sunny@jrstalberto: ..//producer consume... Q Ⓜ ⓘ ⓘ
': 2, 'jeepney': 0, 'bike': 4, 'tryke': 1, 'others': 0}
Preparing message: 275
Message: {'timeuuid_id': '1663266153.16226', 'lgu_code': '120
0', 'sensor_id': 'sensor_07', 'date_saved': '09/16/2022', 'tim
e_saved': '02:22:33', 'total': 7, 'car': 1, 'bus': 0, 'truck':
1, 'jeepney': 0, 'bike': 3, 'tryke': 1, 'others': 1}
Preparing message: 276
Message: {'timeuuid_id': '1663266154.164141', 'lgu_code': '12
00', 'sensor_id': 'sensor_07', 'date_saved': '09/16/2022', 'ti
me_saved': '02:22:34', 'total': 11, 'car': 1, 'bus': 2, 'truck
': 1, 'jeepney': 2, 'bike': 2, 'tryke': 3, 'others': 0}
```

Producer 8

```
+ └ sunny@jrstalberto: ..//producer consume... Q ≡ ○ ○ ○
: 0, 'jeepney': 2, 'bike': 1, 'tryke': 2, 'others': 1}
Preparing message: 280
Message: {'timeuuid_id': '1663266163.056049', 'lgu_code': '12
00', 'sensor_id': 'sensor_08', 'date_saved': '09/16/2022', 'ti
me_saved': '02:22:43', 'total': 11, 'car': 1, 'bus': 2, 'truck
': 2, 'jeepney': 1, 'bike': 4, 'tryke': 1, 'others': 0}
Preparing message: 281
Message: {'timeuuid_id': '1663266164.057859', 'lgu_code': '12
00', 'sensor_id': 'sensor_08', 'date_saved': '09/16/2022', 'ti
me_saved': '02:22:44', 'total': 10, 'car': 0, 'bus': 0, 'truck
': 2, 'jeepney': 2, 'bike': 3, 'tryke': 3, 'others': 0}
```

Producer 9

```
+ └ sunny@jrstalberto: ..//producer consume... Q ≡ ○ ○ ○
: 0, 'jeepney': 2, 'bike': 1, 'tryke': 3, 'others': 1}
Preparing message: 292
Message: {'timeuuid_id': '1663266180.842672', 'lgu_code': '12
00', 'sensor_id': 'sensor_09', 'date_saved': '09/16/2022', 'ti
me_saved': '02:23:00', 'total': 5, 'car': 0, 'bus': 0, 'truck'
: 1, 'jeepney': 1, 'bike': 0, 'tryke': 3, 'others': 0}
Preparing message: 293
Message: {'timeuuid_id': '1663266181.844714', 'lgu_code': '12
00', 'sensor_id': 'sensor_09', 'date_saved': '09/16/2022', 'ti
me_saved': '02:23:01', 'total': 16, 'car': 4, 'bus': 1, 'truck
': 1, 'jeepney': 2, 'bike': 5, 'tryke': 2, 'others': 1}
```

Producer 10

```
+ └ sunny@jrstalberto: ..//producer consume... Q ≡ ○ ○ ○
: 2, 'jeepney': 2, 'bike': 1, 'tryke': 2, 'others': 2}
Preparing message: 294
Message: {'timeuuid_id': '1663266189.479817', 'lgu_code': '12
00', 'sensor_id': 'sensor_10', 'date_saved': '09/16/2022', 'ti
me_saved': '02:23:09', 'total': 7, 'car': 1, 'bus': 2, 'truck'
: 1, 'jeepney': 1, 'bike': 0, 'tryke': 2, 'others': 0}
Preparing message: 295
Message: {'timeuuid_id': '1663266190.481621', 'lgu_code': '12
00', 'sensor_id': 'sensor_10', 'date_saved': '09/16/2022', 'ti
me_saved': '02:23:10', 'total': 15, 'car': 3, 'bus': 1, 'truck
': 1, 'jeepney': 2, 'bike': 4, 'tryke': 2, 'others': 2}
```

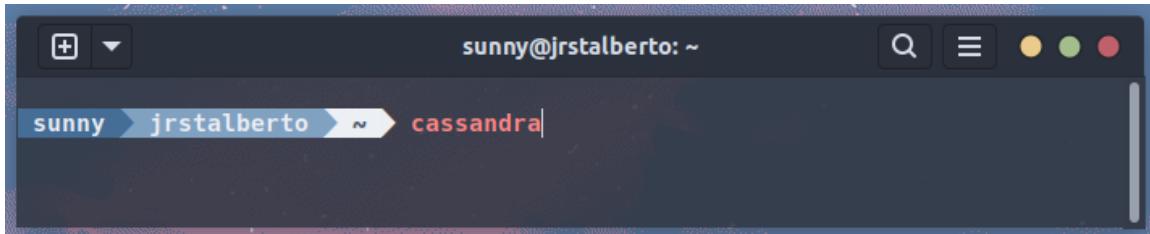
Consumer

```
+ sunny@jrstalberto: .. /producer consume... Q = ⓘ ⓘ ⓘ nsor_id": "sensor_08", "date_saved": "09/16/2022", "time_saved": "02:23:27", "total": 11, "car": 4, "bus": 1, "truck": 1, "jEEPNEY": 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, "jEEPNEY": 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, "jEPNEY": 1, "BIKE": 0, "TRYKE": 0, "OTHERS": 1}'
```

Initialize the Cassandra Environment

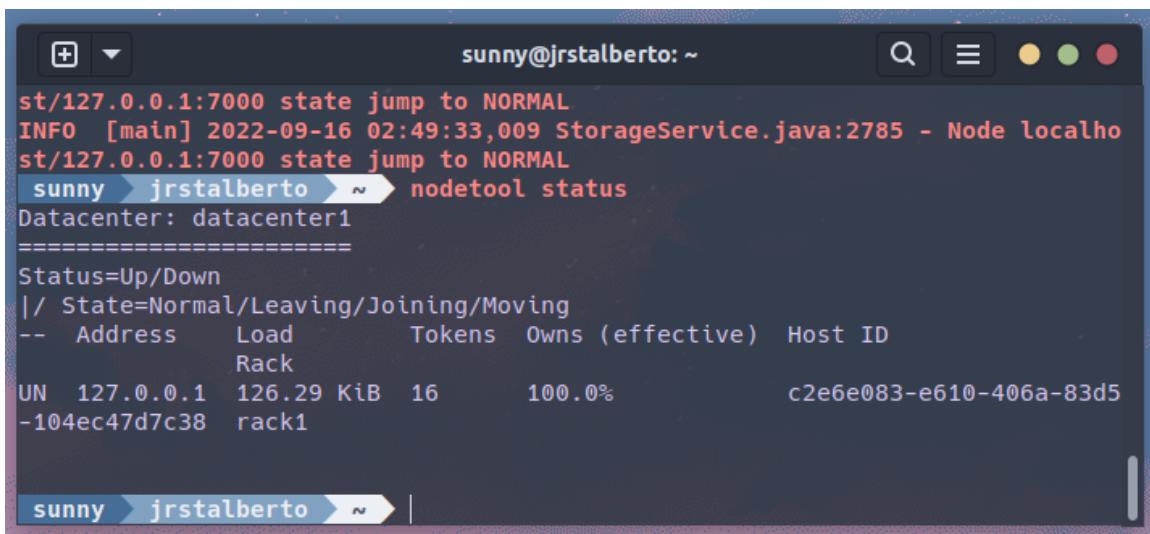
Start the Cassandra environment

```
cassandra
```

A screenshot of a terminal window titled "sunny@jrstalberto: ~". The window has a dark theme with light-colored text. The command "cassandra" is typed into the input field at the bottom. The title bar shows the user's name and host, and there are standard window control buttons (minimize, maximize, close) in the top right corner.

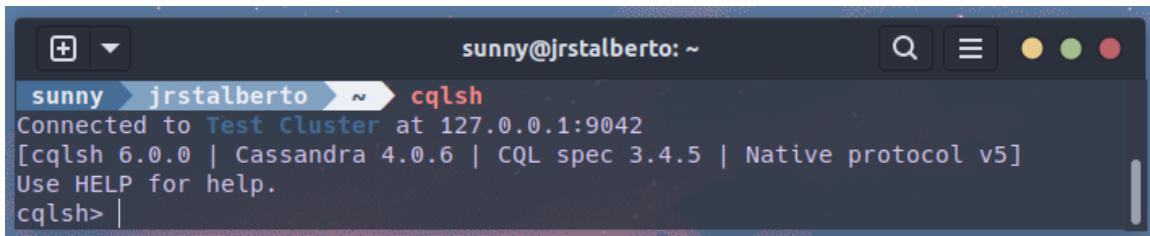
Verify if Cassandra started successfully

```
nodetool status
```

A screenshot of a terminal window titled "sunny@jrstalberto: ~". The window displays the output of the "nodetool status" command. The output shows a single node (UN) in state "Normal" with address 127.0.0.1, load 126.29 KiB, tokens 16, and host ID c2e6e083-e610-406a-83d5. The terminal has a dark theme with light-colored text and standard window controls.

Connect to the database

```
cqlsh
```

A screenshot of a terminal window titled "sunny@jrstalberto: ~". The window shows the connection information for "cqlsh" to a "Test Cluster" at 127.0.0.1:9042. It includes details about the CQL version (3.4.5), Native protocol (v5), and the command prompt "cqlsh>". The terminal has a dark theme with light-colored text and standard window controls.

Create a new keyspace

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

```
sunny@jrstalberto: ~
datascisample    system_auth      system_traces
mysamplekeyspace system_distributed system_views
system          system_schema     system_virtual_schema

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

Warnings :
Your replication factor 3 for keyspace sensor_vehicle is higher than the number of nodes 1

cqlsh> |
```

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.

```
cqlsh> ALTER KEYSPACE sensor_vehicle WITH replication = {'class': 'SimpleStrategy', 'replication_factor' : 1};
```

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)
    ... );
```

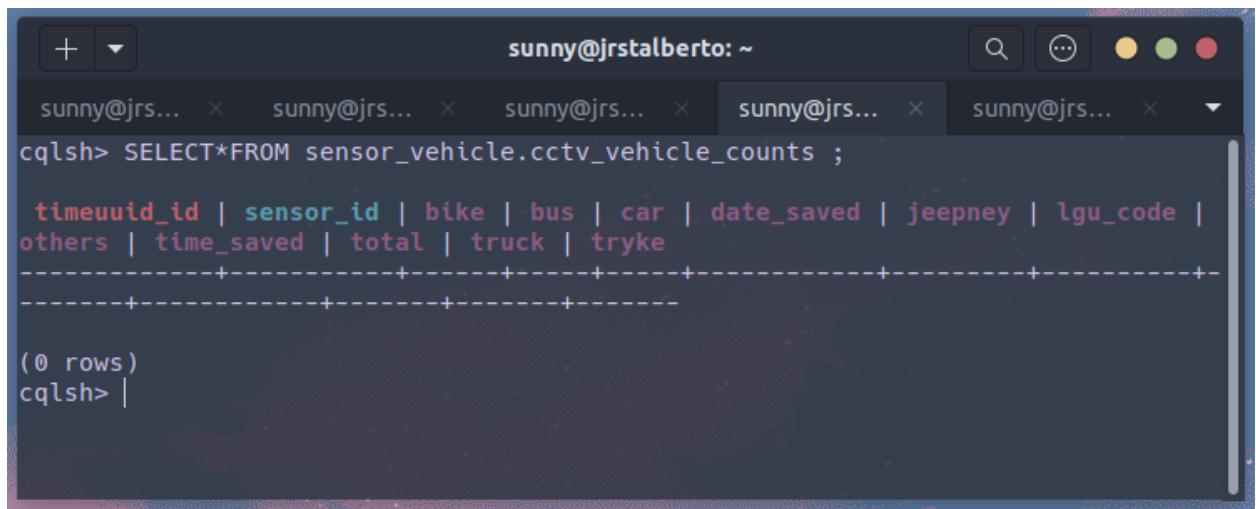


A screenshot of a terminal window titled "sunny@jrstalberto: ~". The window shows a CQLSH session. The user has run the command "use sensor_vehicle" and then "describe tables". A CREATE TABLE statement for "cctv_vehicle_counts" is displayed, defining columns for timeuuid_id, lgu_code, sensor_id, date_saved, time_saved, total, car, bus, truck, jeepney, bike, tryke, and others, with a primary key on (timeuuid_id, sensor_id). The session ends with a final "cqlsh:sensor_vehicle>" prompt.

```
cqlsh> use sensor_vehicle
...
cqlsh:sensor_vehicle> describe tables
cqlsh:sensor_vehicle> 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)
    ... );
cqlsh:sensor_vehicle>
```

Check if the table was created successfully

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



A screenshot of a terminal window titled "sunny@jrstalberto: ~". The user has run the query "SELECT*FROM sensor_vehicle.cctv_vehicle_counts ;". The output shows the schema of the table with columns: timeuuid_id, sensor_id, bike, bus, car, date_saved, jeepney, lgu_code, others, time_saved, total, truck, and tryke. Below the schema, it indicates "(0 rows)" and ends with a final "cqlsh>" prompt.

```
sunny@jrs... × sunny@jrs... × sunny@jrs... × sunny@jrs... × sunny@jrs... × sunny@jrs... ×
cqlsh> SELECT*FROM sensor_vehicle.cctv_vehicle_counts ;
timeuuid_id | sensor_id | bike | bus | car | date_saved | jeepney | lgu_code |
others | time_saved | total | truck | tryke
-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+
(0 rows)
cqlsh>
```

Initialize the Spark Environment

Prepare the code for the stream handler

```
stweamhandler.py > ...
1 import findspark
2 findspark.init()
3
4 from pyspark.sql import SparkSession
5 from pyspark.sql.functions import *
6 from pyspark.sql.types import *
7
8 cassandra_host_name = "localhost"
9 cassandra_port_no = "9042"
10 cassandra_keyspace_name = "sensor_vehicle"
11 cassandra_table_name = "cctv_vehicle_counts"
12
13 def save_to_cassandra(current_df, epoch_id):
14     print("Printing epoch_id: ")
15     print(epoch_id)
16
17     print("Printing before Cassandra Table save: " + str(epoch_id))
18     current_df \
19         .write \
20             .format("org.apache.spark.sql.cassandra") \
21             .mode("append") \
22             .options(table=cassandra_table_name, keyspace=cassandra_keyspace_name) \
23             .save()
24
25 if __name__ == "__main__":
26     spark = SparkSession \
27         .builder \
28         .appName("StreamHandler") \
29         .config('spark.cassandra.connection.host', cassandra_host_name) \
30         .config('spark.cassandra.connection.port', cassandra_port_no) \
31         .getOrCreate()
32
33 schema = StructType([
34     StructField("timeuuid_id", StringType(), True),
35     StructField("lgu_code", StringType(), True),
36     StructField("sensor_id", StringType(), True),
37     StructField("date_saved", StringType(), True),
38     StructField("time_saved", StringType(), True),
39     StructField("total", DecimalType(), True),
40     StructField("car", DecimalType(), True),
41     StructField("bus", DecimalType(), True),
42     StructField("truck", DecimalType(), True),
43     StructField("jeepney", DecimalType(), True),
44     StructField("bike", DecimalType(), True),
45     StructField("tryke", DecimalType(), True),
46     StructField("others", DecimalType(), True)
47 ])
48
49 lines = spark \
50     .readStream \
51     .format("kafka") \
52     .option("kafka.bootstrap.servers", "localhost:9092") \
53     .option("subscribe", "cctv_vehicle_counts") \
54     .option("startingOffsets", "latest") \
55     .load() \
56     .select(from_json(col("value").cast("string"), schema).alias("parsed_value")).select(col("parsed_value"))
57
58 query = lines \
59     .writeStream \
60     .trigger(processingTime='15 seconds') \
61     .outputMode("update") \
62     .foreachBatch(save_to_cassandra) \
63     .start()
64
65 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-c
assandra-connector_2.12:3.2.0 stweamhandler.py
```

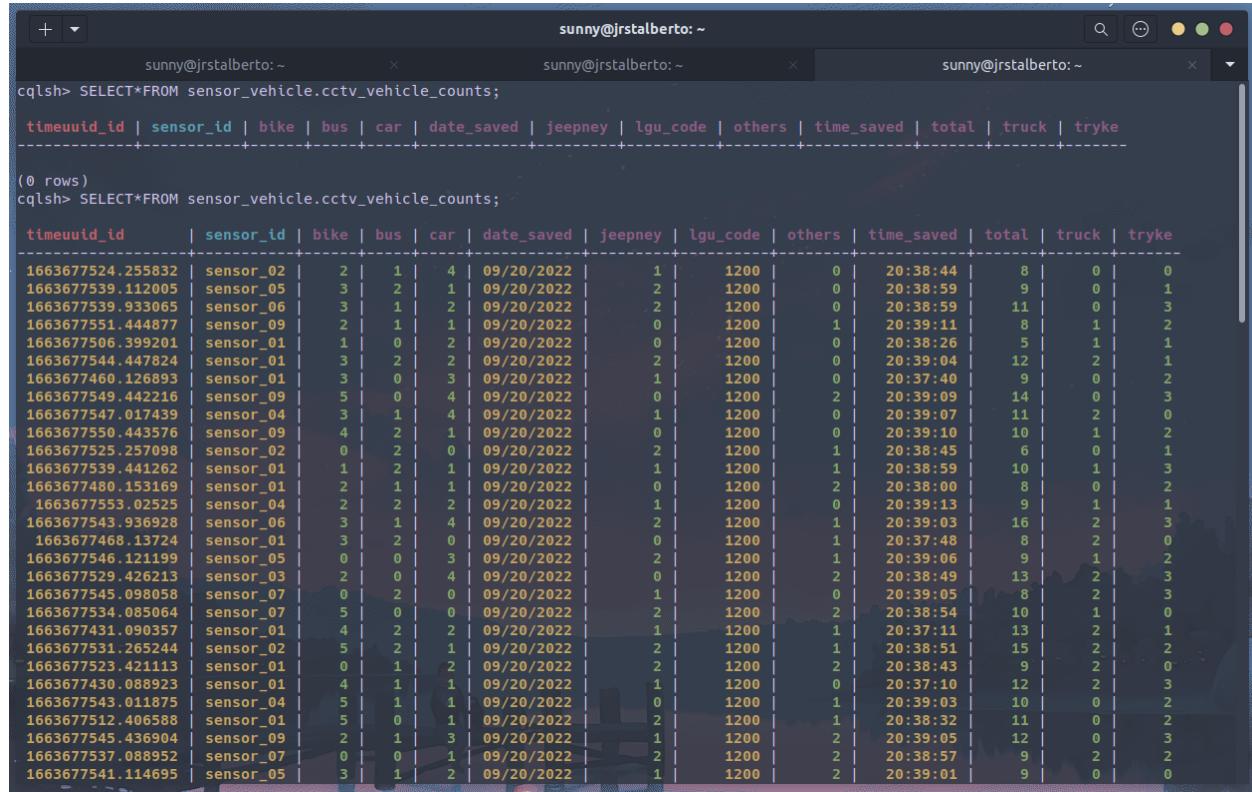


The screenshot shows a terminal window titled "sunny@jrstalberto: ~/Downloads". The command entered is:

```
sunny > jrstalberto > cd Downloads
sunny > jrstalberto > ~/Downloads > 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;
```



The screenshot shows a terminal window titled "sunny@jrstalberto: ~" with three tabs. The first tab shows the query:

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

The second tab shows the schema:

```
timeuuid_id | sensor_id | bike | bus | car | date_saved | jeepney | lgu_code | others | time_saved | total | truck | tryke
```

(0 rows)

The third tab shows the actual data:

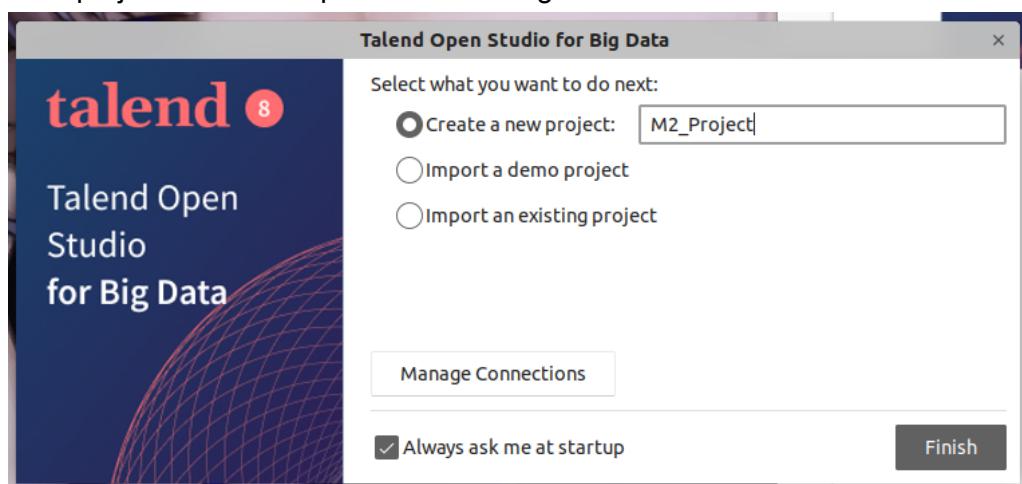
| timeuuid_id | sensor_id | bike | bus | car | date_saved | jeepney | lgu_code | others | time_saved | total | truck | tryke |
|-------------------|-----------|------|-----|-----|------------|---------|----------|--------|------------|-------|-------|-------|
| 1663677524.255832 | sensor_02 | 2 | 1 | 4 | 09/20/2022 | 1 | 1200 | 0 | 20:38:44 | 8 | 0 | 0 |
| 1663677539.112005 | sensor_05 | 3 | 2 | 1 | 09/20/2022 | 2 | 1200 | 0 | 20:38:59 | 9 | 0 | 1 |
| 1663677539.933065 | sensor_06 | 3 | 1 | 2 | 09/20/2022 | 2 | 1200 | 0 | 20:38:59 | 11 | 0 | 3 |
| 1663677551.444877 | sensor_09 | 2 | 1 | 1 | 09/20/2022 | 0 | 1200 | 1 | 20:39:11 | 8 | 1 | 2 |
| 1663677506.399201 | sensor_01 | 1 | 0 | 2 | 09/20/2022 | 0 | 1200 | 0 | 20:38:26 | 5 | 1 | 1 |
| 1663677544.447824 | sensor_01 | 3 | 2 | 2 | 09/20/2022 | 2 | 1200 | 0 | 20:39:04 | 12 | 2 | 1 |
| 1663677460.126893 | sensor_01 | 3 | 0 | 3 | 09/20/2022 | 1 | 1200 | 0 | 20:37:40 | 9 | 0 | 2 |
| 1663677549.442216 | sensor_09 | 5 | 0 | 4 | 09/20/2022 | 0 | 1200 | 2 | 20:39:09 | 14 | 0 | 3 |
| 1663677547.017439 | sensor_04 | 3 | 1 | 4 | 09/20/2022 | 1 | 1200 | 0 | 20:39:07 | 11 | 2 | 0 |
| 1663677550.443576 | sensor_09 | 4 | 2 | 1 | 09/20/2022 | 0 | 1200 | 0 | 20:39:10 | 10 | 1 | 2 |
| 1663677525.257098 | sensor_02 | 0 | 2 | 0 | 09/20/2022 | 2 | 1200 | 1 | 20:38:45 | 6 | 0 | 1 |
| 1663677539.441262 | sensor_01 | 1 | 2 | 1 | 09/20/2022 | 1 | 1200 | 1 | 20:38:59 | 10 | 1 | 3 |
| 1663677480.153169 | sensor_01 | 2 | 1 | 1 | 09/20/2022 | 0 | 1200 | 2 | 20:38:00 | 8 | 0 | 2 |
| 1663677553.02525 | sensor_04 | 2 | 2 | 2 | 09/20/2022 | 1 | 1200 | 0 | 20:39:13 | 9 | 1 | 1 |
| 1663677543.936928 | sensor_06 | 3 | 1 | 4 | 09/20/2022 | 2 | 1200 | 1 | 20:39:03 | 16 | 2 | 3 |
| 1663677468.13724 | sensor_01 | 3 | 2 | 0 | 09/20/2022 | 0 | 1200 | 1 | 20:37:48 | 8 | 2 | 0 |
| 1663677546.121199 | sensor_05 | 0 | 0 | 3 | 09/20/2022 | 2 | 1200 | 1 | 20:39:06 | 9 | 1 | 2 |
| 1663677529.426213 | sensor_03 | 2 | 0 | 4 | 09/20/2022 | 0 | 1200 | 2 | 20:38:49 | 13 | 2 | 3 |
| 1663677545.098058 | sensor_07 | 0 | 2 | 0 | 09/20/2022 | 1 | 1200 | 0 | 20:39:05 | 8 | 2 | 3 |
| 1663677534.085064 | sensor_07 | 5 | 0 | 0 | 09/20/2022 | 2 | 1200 | 2 | 20:38:54 | 10 | 1 | 0 |
| 1663677431.090357 | sensor_01 | 4 | 2 | 2 | 09/20/2022 | 1 | 1200 | 1 | 20:37:11 | 13 | 2 | 1 |
| 1663677531.265244 | sensor_02 | 5 | 2 | 1 | 09/20/2022 | 2 | 1200 | 1 | 20:38:51 | 15 | 2 | 2 |
| 1663677523.421113 | sensor_01 | 0 | 1 | 2 | 09/20/2022 | 2 | 1200 | 2 | 20:38:43 | 9 | 2 | 0 |
| 1663677430.088923 | sensor_01 | 4 | 1 | 1 | 09/20/2022 | 1 | 1200 | 0 | 20:37:10 | 12 | 2 | 3 |
| 1663677543.011875 | sensor_04 | 5 | 1 | 1 | 09/20/2022 | 0 | 1200 | 1 | 20:39:03 | 10 | 0 | 2 |
| 1663677512.406588 | sensor_01 | 5 | 0 | 1 | 09/20/2022 | 2 | 1200 | 1 | 20:38:32 | 11 | 0 | 2 |
| 1663677545.436904 | sensor_09 | 2 | 1 | 3 | 09/20/2022 | 1 | 1200 | 2 | 20:39:05 | 12 | 0 | 3 |
| 1663677537.088952 | sensor_07 | 0 | 0 | 1 | 09/20/2022 | 2 | 1200 | 2 | 20:38:57 | 9 | 2 | 2 |
| 1663677541.114695 | sensor_05 | 3 | 1 | 2 | 09/20/2022 | 1 | 1200 | 2 | 20:39:01 | 9 | 0 | 0 |

Start Cassandra

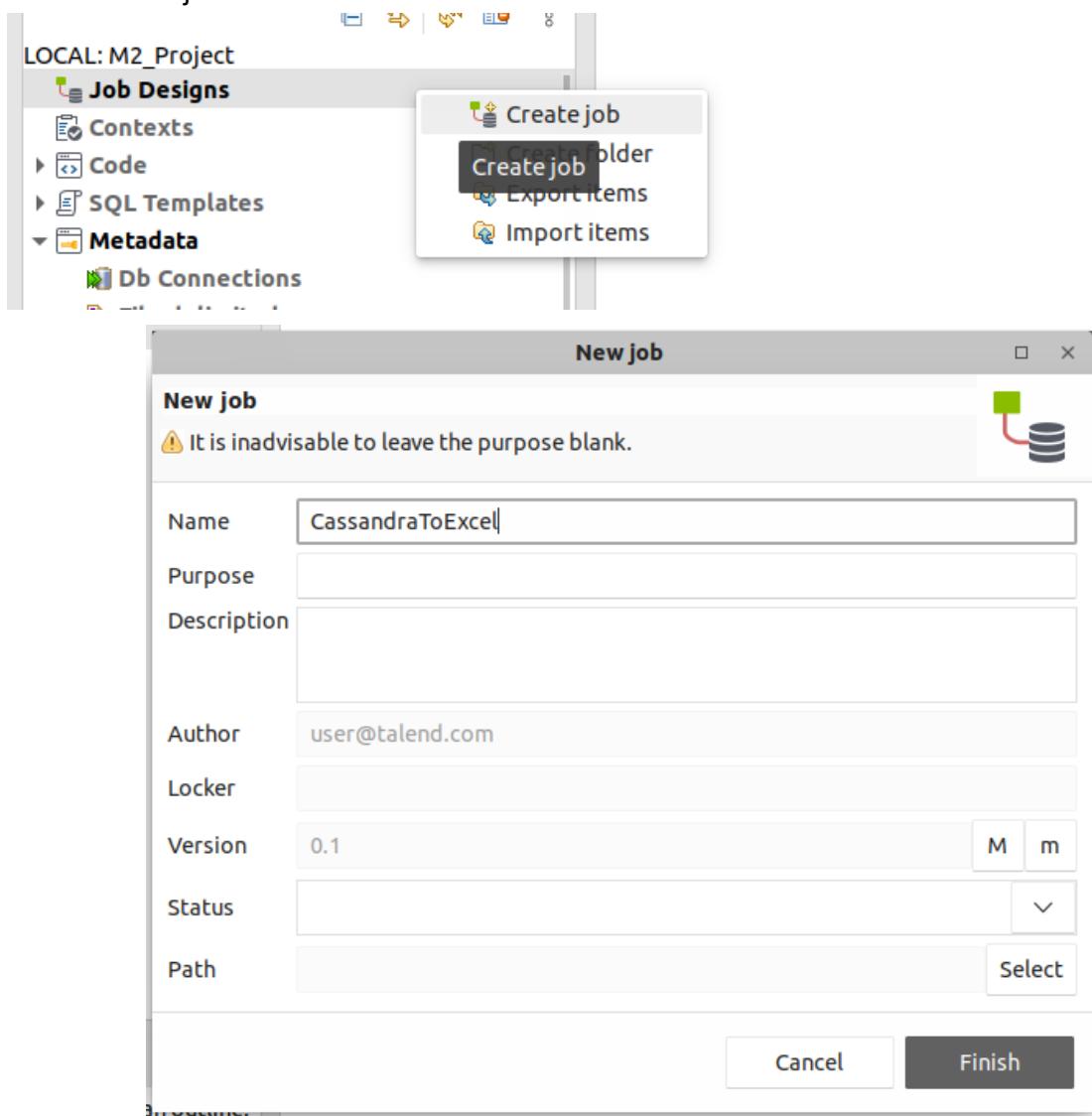
```
. sunny > jrstalberto > cassandra

sunny > jrstalberto > CompilerOracle: dontinline org/apache/cassandra/db/Columns$Serializer.deserializeLargeSubset (Lorg/apache/cassandra/io/util/DataInputPlus;Lorg/apache/cassandra/db/Columns;ILorg/apache/cassandra/db/Columns;
CompilerOracle: dontinline org/apache/cassandra/db/Columns$Serializer.serializeLargeSubset (Ljava/util/Collection;ILorg/apache/cassandra/db/Columns;ILorg/apache/cassandra/io/util/DataOutputPlus;)V
CompilerOracle: dontinline org/apache/cassandra/db/Columns$Serializer.serializeLargeSubsetSize (Ljava/util/Collection;ILorg/apache/cassandra/db/Columns;I)I
CompilerOracle: dontinline org/apache/cassandra/db/commitlog/AbstractCommitLogSegmentManager.advanceAllocatingFrom (Lorg/apache/cassandra/db/commitlog/CommitLogSegment;)V
CompilerOracle: dontinline org/apache/cassandra/db/transform/BaseIterator.tryGetMoreContents ()Z
CompilerOracle: dontinline org/apache/cassandra/db/transform/StoppingTransformation.stop ()V
CompilerOracle: dontinline org/apache/cassandra/db/transform/StoppingTransformation.stopInPartition ()V
CompilerOracle: dontinline org/apache/cassandra/io/util/BufferedDataOutputStreamPlus.doFlush (I)V
CompilerOracle: dontinline org/apache/cassandra/io/util/BufferedDataOutputStreamPlus.writeSlow (JI)V
CompilerOracle: dontinline org/apache/cassandra/io/util/RebufferingInputStream.readPrimitiveSlowly (I)J
CompilerOracle: exclude org/apache/cassandra/utils/JVMStabilityInspector.forceHeapSpaceOomMaybe (Ljava/lang/OutOfMemoryError;)V
```

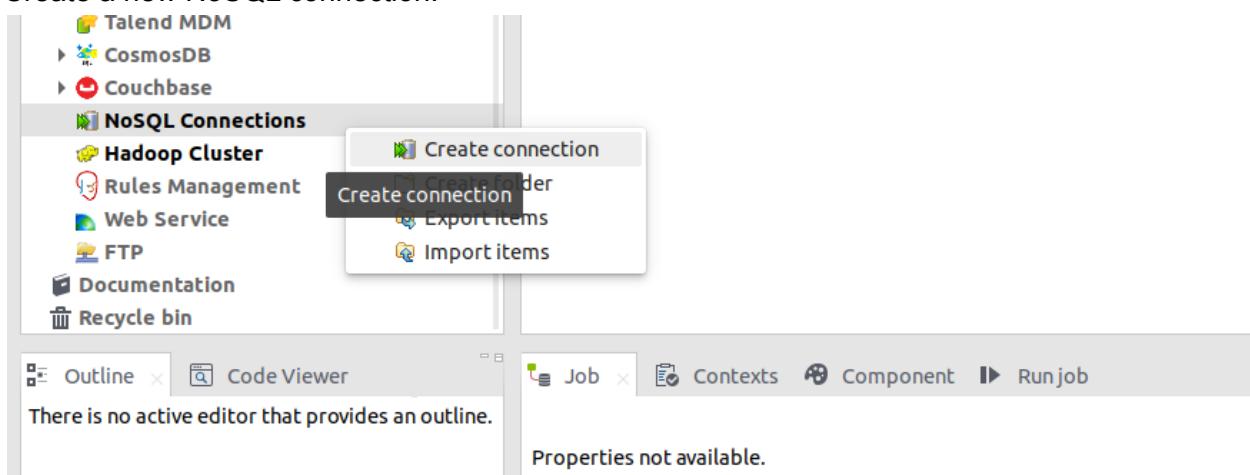
Create a new project in Talend Open Studio for Big Data



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

The screenshot shows a software interface for managing NoSQL connections and schemas.

Left Sidebar:

- Couchbase
- NoSQL Connections
 - NoSQLConnToCassandra 0.1
 - Hadoop Cluster
 - Rules Management
 - Web Service
 - FTP
 - Documentation
 - Recycle bin

Central Area (Top):

A context menu is open over the connection "NoSQLConnToCassandra 0.1". The menu items are:

- Edit connection
- Retrieve Schema
- Detected
- Delete
- Copy
- Duplicate
- Export items

The "Retrieve Schema" item is highlighted.

Central Area (Bottom):

A window titled "Schema" is open, showing the "New Schema on 'NoSQLConnToCassandra'" screen.

Schema Selection:

Select schema to create
Name Filter: cctv_vehicle_counts

| Name | Type | Column Number | Creation Status |
|---------------------|---------------|---------------|-----------------|
| cctv_vehicle_counts | Column Family | 13 | Success |

Schema Definition:

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 ▾

Schema:

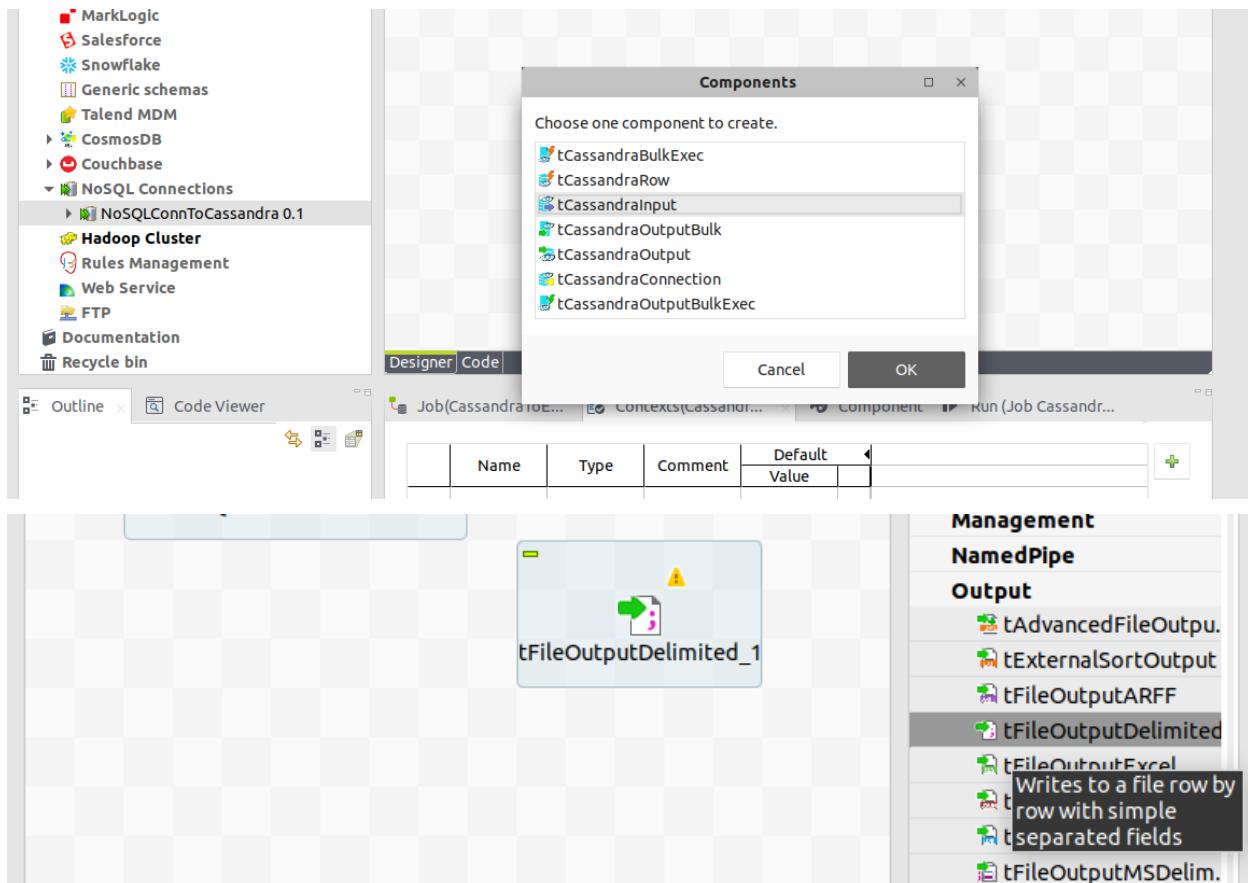
| Column | Db Column | Key | Type | DB Type | Null | Date Patter | Length | Precision | Default | Comment |
|-------------|-------------|--------------------------|---------|---------|-------------------------------------|-------------|--------|-----------|---------|---------|
| timeuuid_id | timeuuid_id | <input type="checkbox"/> | String | text | <input checked="" type="checkbox"/> | | 0 | 0 | | |
| sensor_id | sensor_id | <input type="checkbox"/> | String | text | <input checked="" type="checkbox"/> | | 0 | 0 | | |
| bike | bike | <input type="checkbox"/> | Integer | int | <input checked="" type="checkbox"/> | | 0 | 0 | | |
| bus | bus | <input type="checkbox"/> | Integer | int | <input checked="" type="checkbox"/> | | 0 | 0 | | |
| car | car | <input type="checkbox"/> | Integer | int | <input checked="" type="checkbox"/> | | 0 | 0 | | |
| date_saved | date_saved | <input type="checkbox"/> | String | text | <input checked="" type="checkbox"/> | | 0 | 0 | | |
| jeepney | jeepney | <input type="checkbox"/> | Integer | int | <input checked="" type="checkbox"/> | | 0 | 0 | | |
| lgu_code | lgu_code | <input type="checkbox"/> | String | text | <input checked="" type="checkbox"/> | | 0 | 0 | | |
| others | others | <input type="checkbox"/> | Integer | int | <input checked="" type="checkbox"/> | | 0 | 0 | | |
| time_saved | time_saved | <input type="checkbox"/> | String | text | <input checked="" type="checkbox"/> | | 0 | 0 | | |
| total | total | <input type="checkbox"/> | Integer | int | <input checked="" type="checkbox"/> | | 0 | 0 | | |
| truck | truck | <input type="checkbox"/> | Integer | int | <input checked="" type="checkbox"/> | | 0 | 0 | | |
| tryke | tryke | <input type="checkbox"/> | Integer | int | <input checked="" type="checkbox"/> | | 0 | 0 | | |

Buttons at the bottom of the schema definition window:

- Add Schema
- Remove Schema
-
-
-
-
-
-
-
-

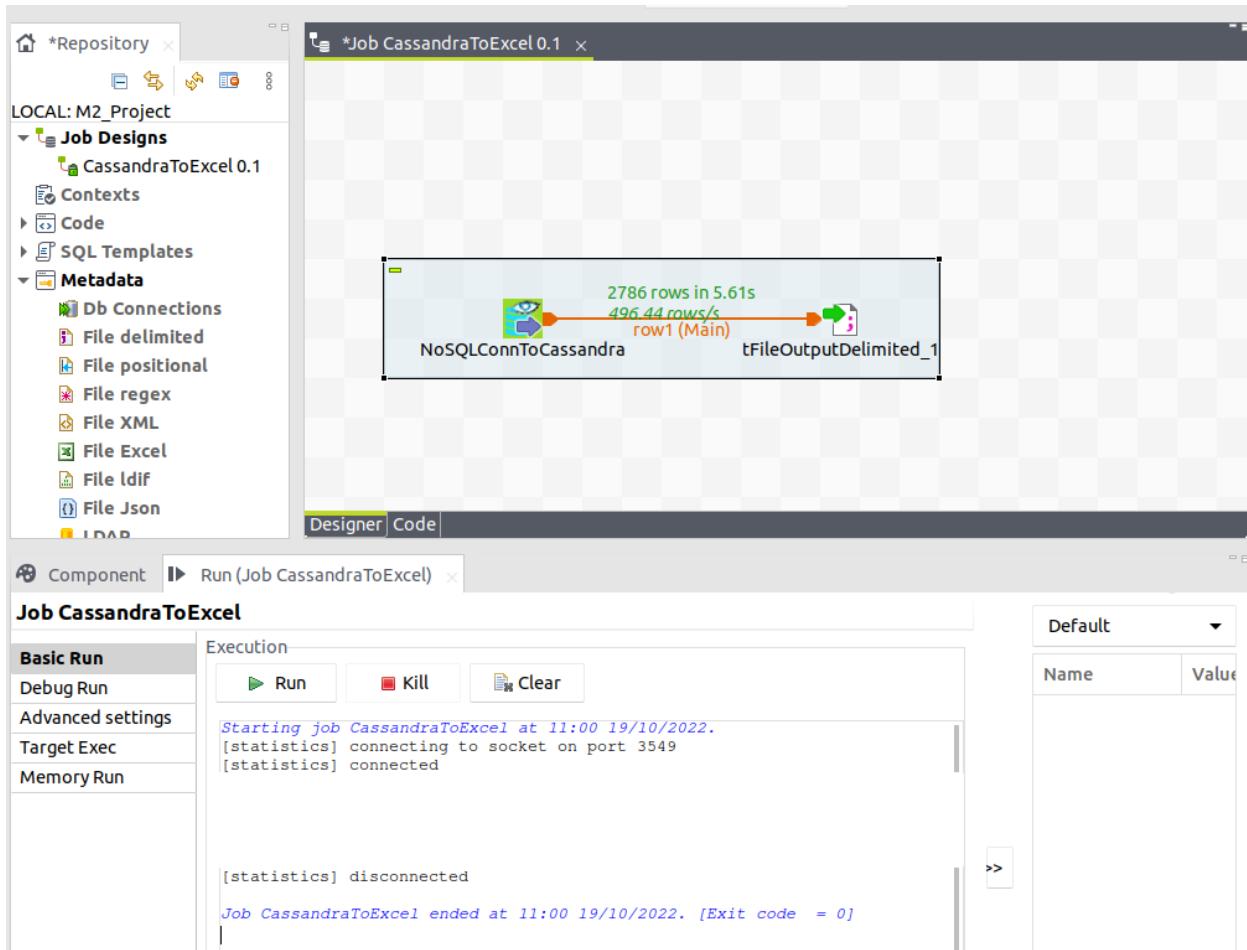
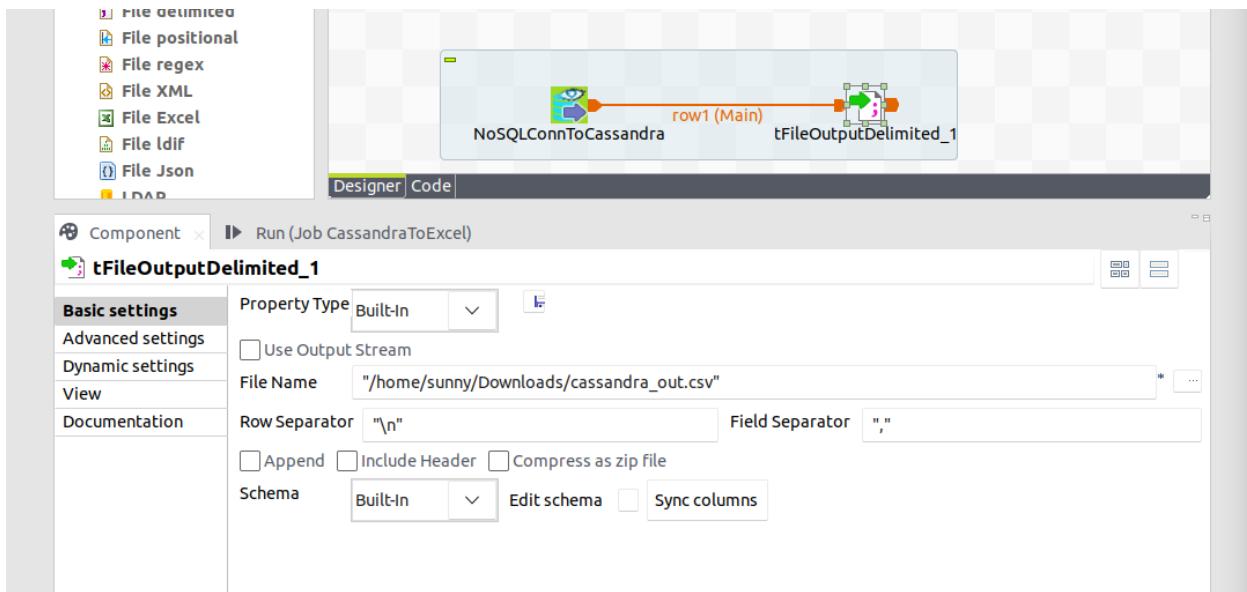
Bottom Navigation:

< Back Next > Cancel Finish



The screenshot shows the configuration of the 'NoSQLConnToCassandra(tCassandraInput_1)' component. The 'Basic settings' tab is active, displaying the following configuration:

- Host:** localhost
- Port:** 9042
- DB Version:** Cassandra 3.0.x
- Keyspace:** sensor_vehicle
- Query:** select * from sensor_vehicle.cctv_vehicle_counts



| | cassandra_out.csv | ~Downloads |
|----|---|------------|
| 1 | 1663677591.500925,sensor_03,5,1,0,09/20/2022,2,1200,0,20:39:51,12,1,3 | |
| 2 | 1663677753.707657,sensor_03,1,2,0,09/20/2022,1,1200,0,20:42:33,5,1,0 | |
| 3 | 1663677584.331852,sensor_02,4,1,3,09/20/2022,2,1200,0,20:39:44,12,0,2 | |
| 4 | 1663677758.366493,sensor_07,1,0,1,09/20/2022,0,1200,1,20:42:38,3,0,0 | |
| 5 | 1663677728.682804,sensor_01,0,1,0,09/20/2022,0,1200,1,20:42:08,5,0,3 | |
| 6 | 1663677613.784032,sensor_08,1,0,0,09/20/2022,2,1200,2,20:40:13,7,0,2 | |
| 7 | 1663677718.917363,sensor_08,2,2,4,09/20/2022,1,1200,2,20:41:58,14,2,1 | |
| 8 | 1663677748.270467,sensor_04,1,0,3,09/20/2022,1,1200,1,20:42:28,6,0,0 | |
| 9 | 1663677772.22643,sensor_06,5,2,2,09/20/2022,0,1200,0,20:42:52,13,2,2 | |
| 10 | 1663677653.232294,sensor_07,4,2,3,09/20/2022,1,1200,1,20:40:53,12,1,0 | |
| 11 | 1663677524.255832,sensor_02,2,1,4,09/20/2022,1,1200,0,20:38:44,8,0,0 | |
| 12 | 1663677750.703816,sensor_03,1,0,4,09/20/2022,0,1200,0,20:42:30,7,1,1 | |
| 13 | 1663677721.667752,sensor_09,2,2,4,09/20/2022,1,1200,1,20:42:01,11,1,0 | |
| 14 | 1663677579.490509,sensor_01,4,1,0,09/20/2022,1,1200,0,20:39:39,6,0,0 | |
| 15 | 1663677539.112005,sensor_05,3,2,1,09/20/2022,2,1200,0,20:38:59,9,0,1 | |
| 16 | 1663677593.49698,sensor_09,5,1,2,09/20/2022,0,1200,0,20:39:53,12,1,3 | |
| 17 | 1663677794.250109,sensor_06,5,1,0,09/20/2022,2,1200,1,20:43:14,11,2,0 | |
| 18 | 1663677777.584723,sensor_02,0,2,2,09/20/2022,2,1200,2,20:42:57,9,0,1 | |
| 19 | 1663677594.3451,sensor_02,3,2,2,09/20/2022,0,1200,1,20:39:54,13,2,3 | |
| 20 | 1663677772.731981,sensor_09,2,0,2,09/20/2022,2,1200,0,20:42:52,6,0,0 | |
| 21 | 1663677685.273409,sensor_07,5,1,2,09/20/2022,2,1200,0,20:41:25,12,0,2 | |
| 22 | 1663677612.205252,sensor_05,2,2,0,09/20/2022,2,1200,2,20:40:12,12,2,2 | |
| 23 | 1663677539.933065,sensor_06,3,1,2,09/20/2022,2,1200,0,20:38:59,11,0,3 | |
| 24 | 1663677707.491987,sensor_02,2,1,2,09/20/2022,2,1200,0,20:41:47,8,1,0 | |
| 25 | 1663677646.825225,sensor_08,0,0,0,09/20/2022,2,1200,2,20:40:46,5,0,1 | |
| 26 | 1663677564.461142,sensor_09,4,1,2,09/20/2022,0,1200,1,20:39:24,11,2,1 | |
| 27 | 1663677563.459898,sensor_09,2,0,1,09/20/2022,2,1200,0,20:39:23,5,0,0 | |
| 28 | 1663677748.194737,sensor_06,4,2,2,09/20/2022,0,1200,1,20:42:28,11,1,1 | |
| 29 | 1663677627.54794,sensor_03,0,1,2,09/20/2022,0,1200,1,20:40:27,6,2,0 | |
| 30 | 1663677657.839964,sensor_08,3,0,3,09/20/2022,0,1200,0,20:40:57,11,2,3 | |
| 31 | 1663677551.444877,sensor_09,2,1,1,09/20/2022,0,1200,1,20:39:11,8,1,2 | |
| 32 | 1663677668.251825,sensor_07,0,1,1,09/20/2022,1,1200,0,20:41:08,6,1,2 | |
| 33 | 1663677556.133394,sensor_05,2,2,4,09/20/2022,0,1200,0,20:39:16,13,2,3 | |
| 34 | 1663677691.121263,sensor_06,3,1,2,09/20/2022,1,1200,1,20:41:31,11,1,2 | |

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

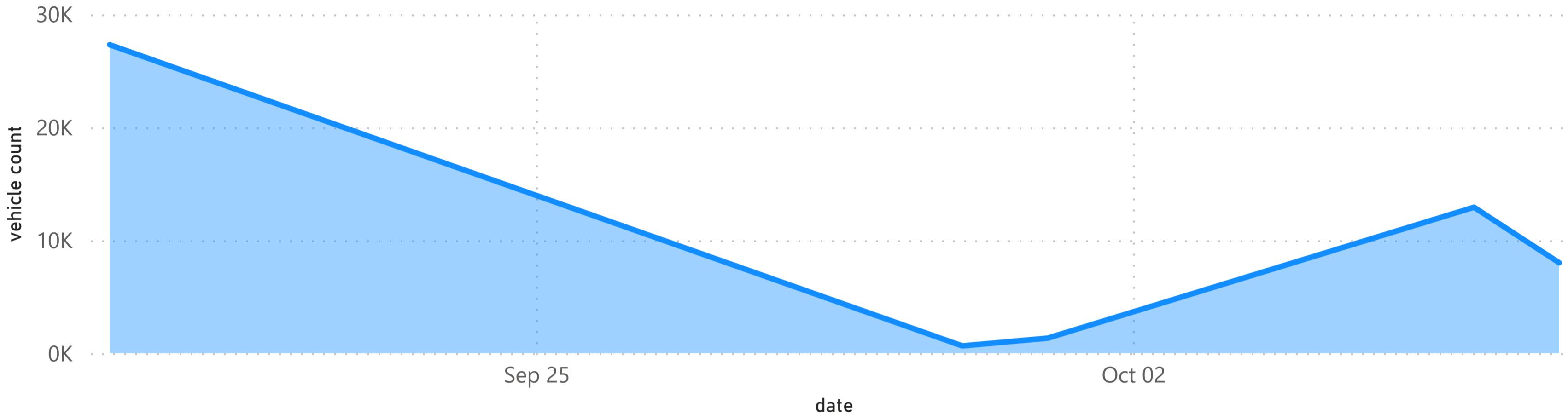


Quick summary of vehicle sensor counts

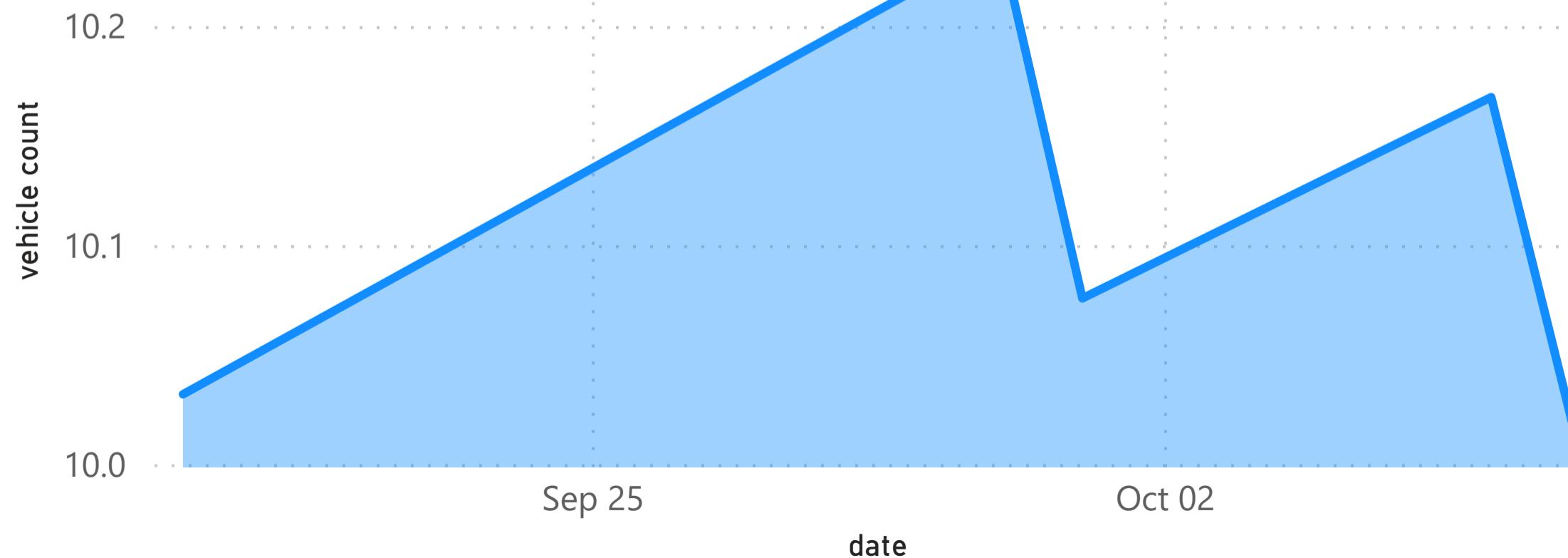
50219

Total vehicles counted

Vehicle counts over time

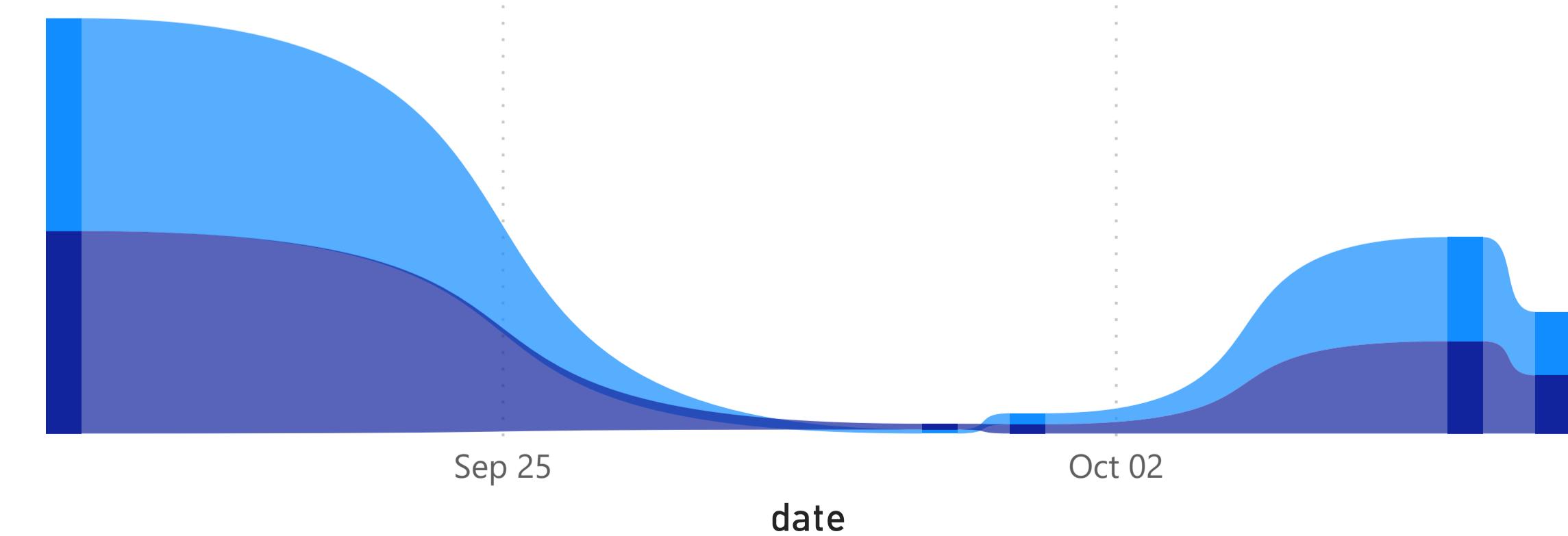


Average vehicle count per day



Vehicles counted by action over time

action ● in ● out



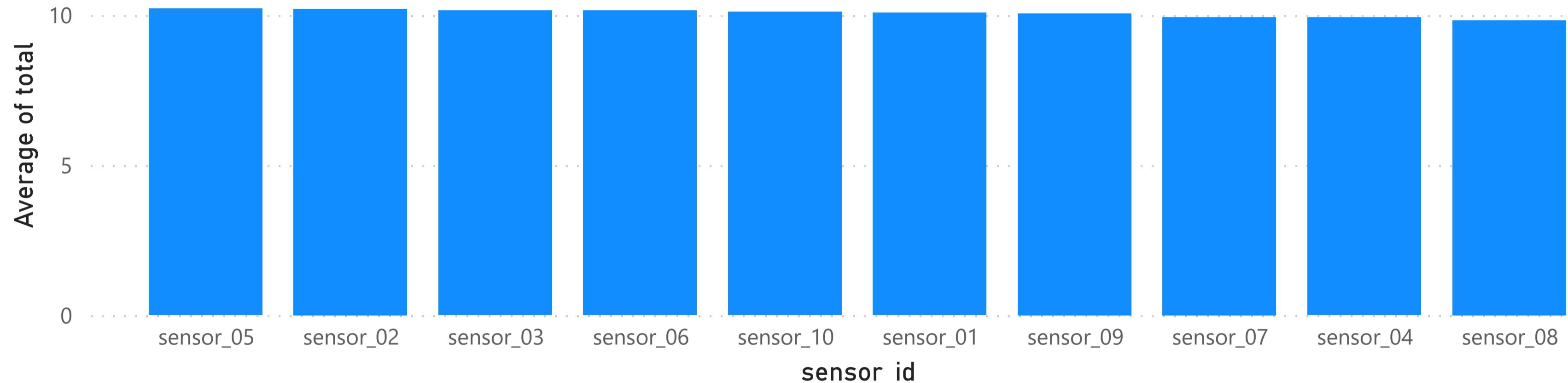


Quick summary of vehicle sensor counts

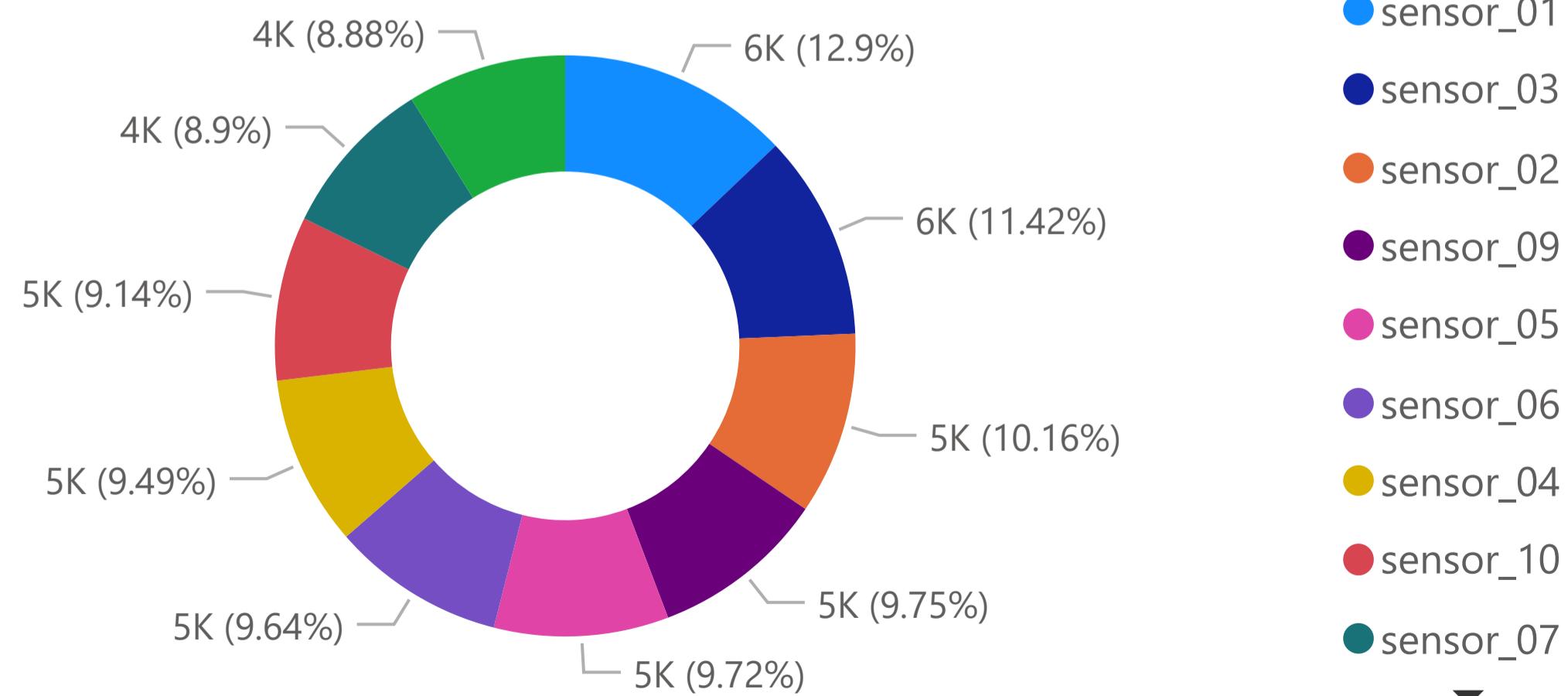
50219

Total vehicles counted

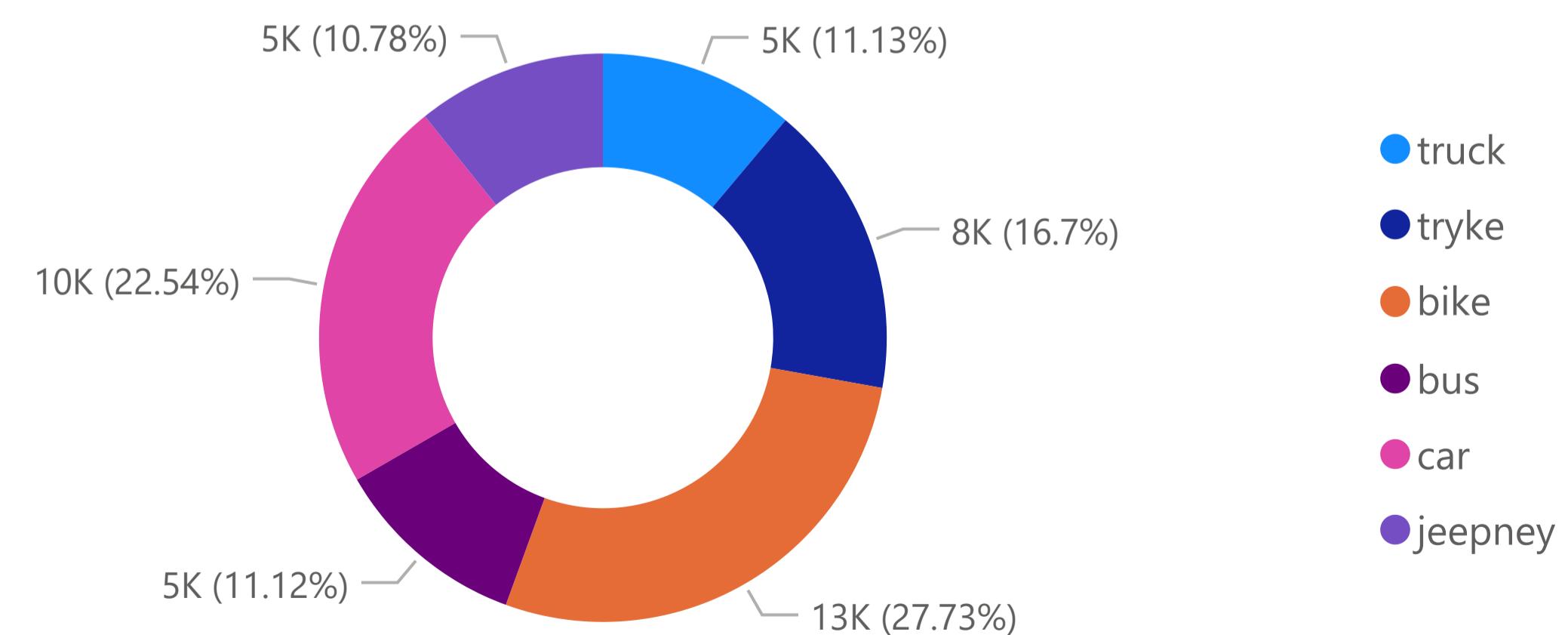
Average vehicle count per sensor



Total vehicle count per sensor



Vehicle type count



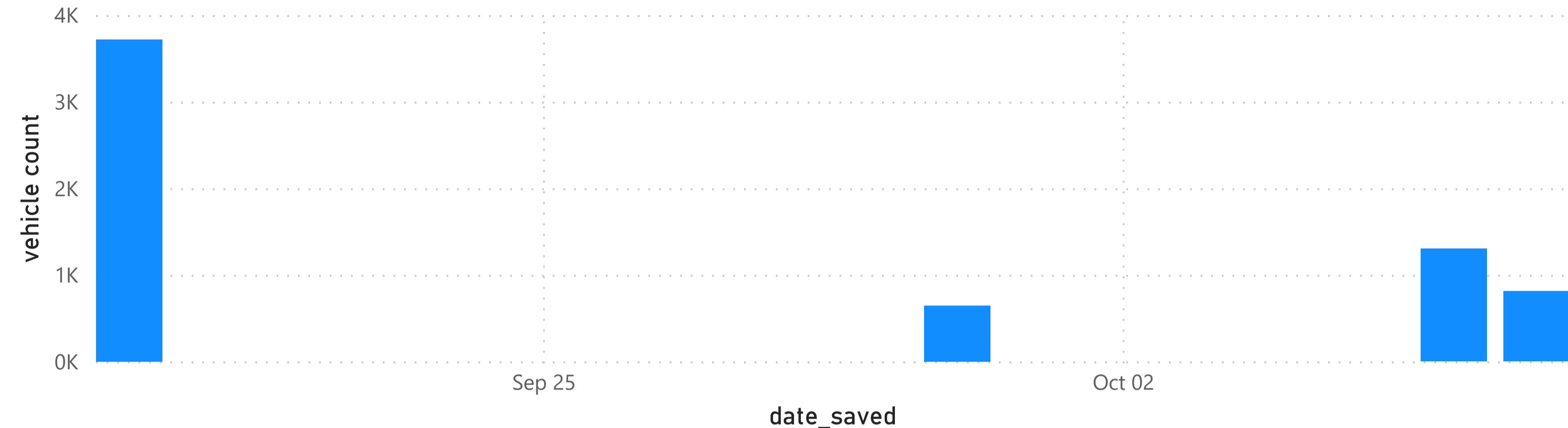


Sensor 1

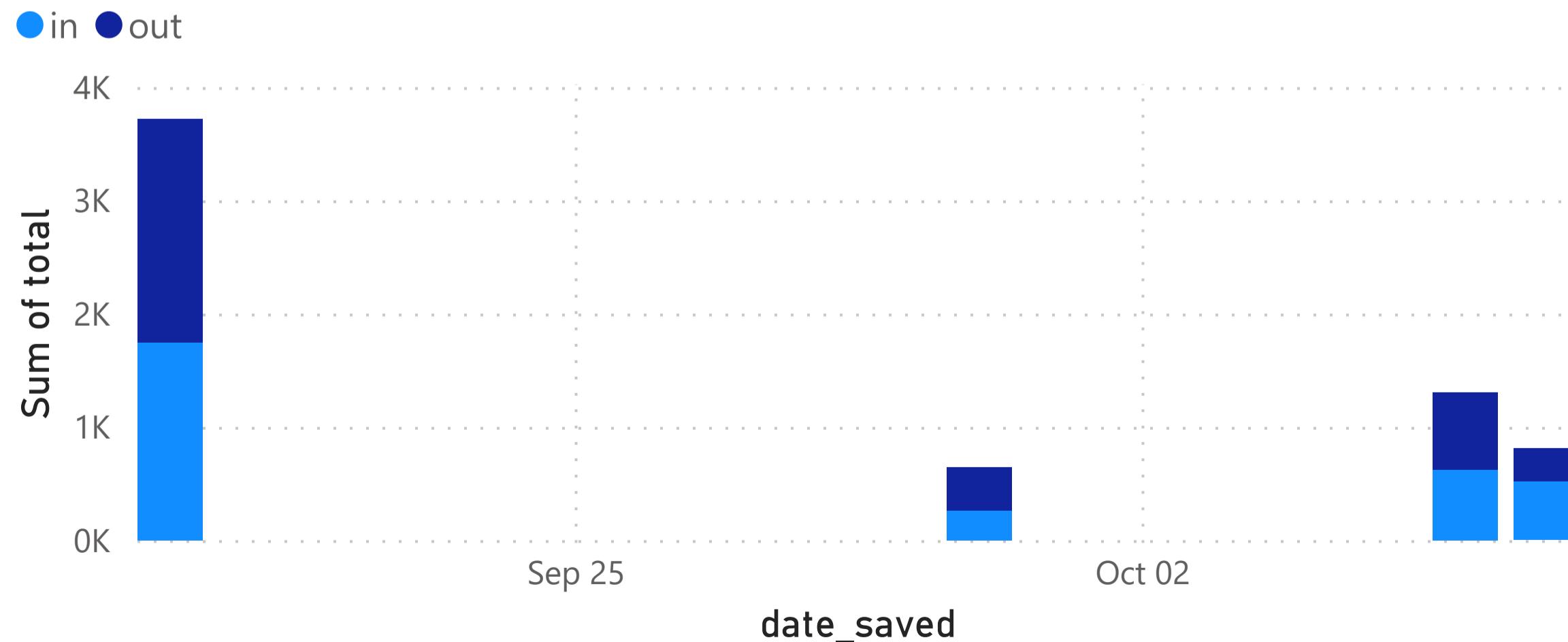
6480

Total vehicles counted

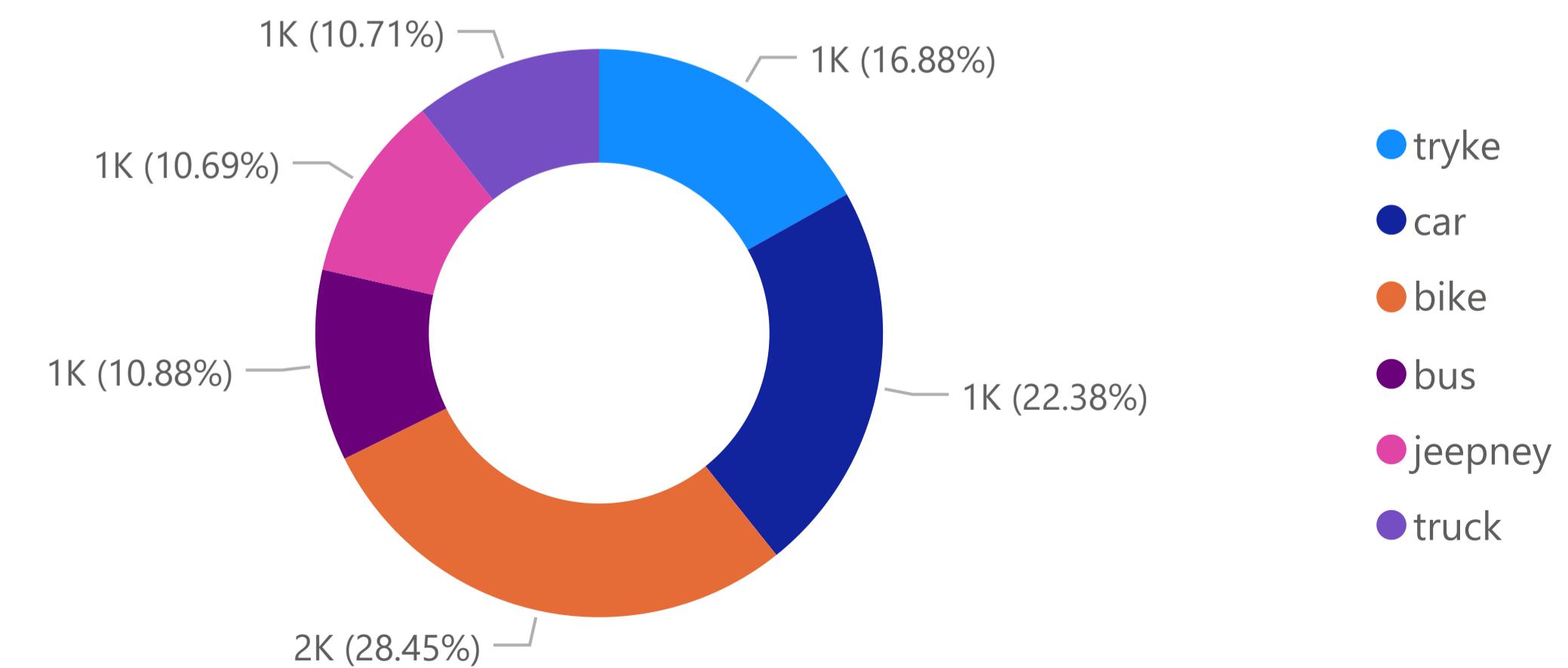
Vehicle count per day



Vehicle actions per day



Vehicle type count for sensor 1



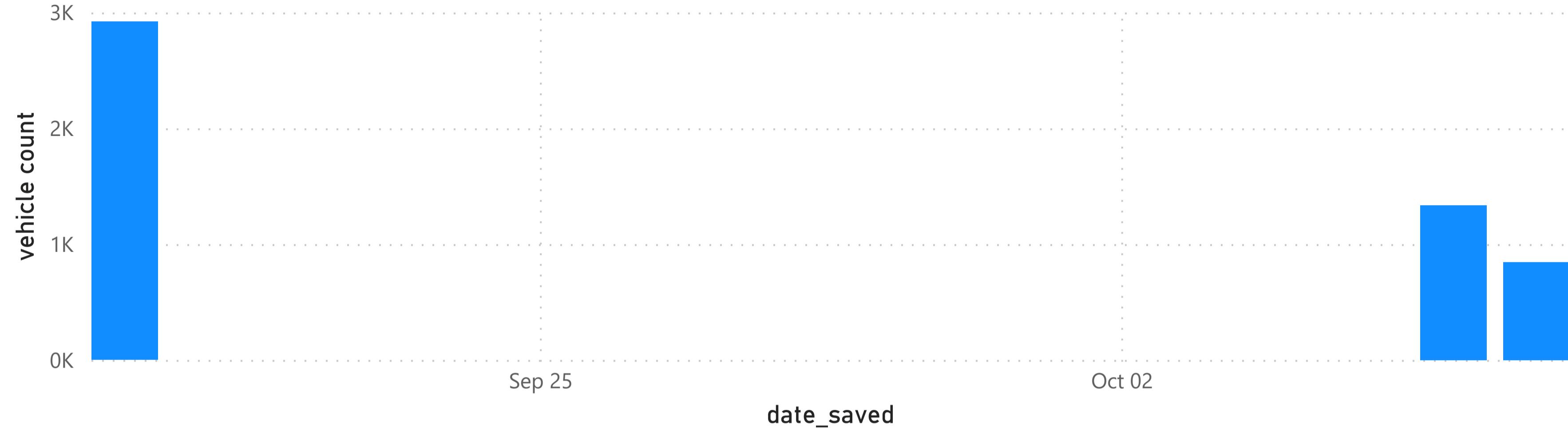


Sensor 2

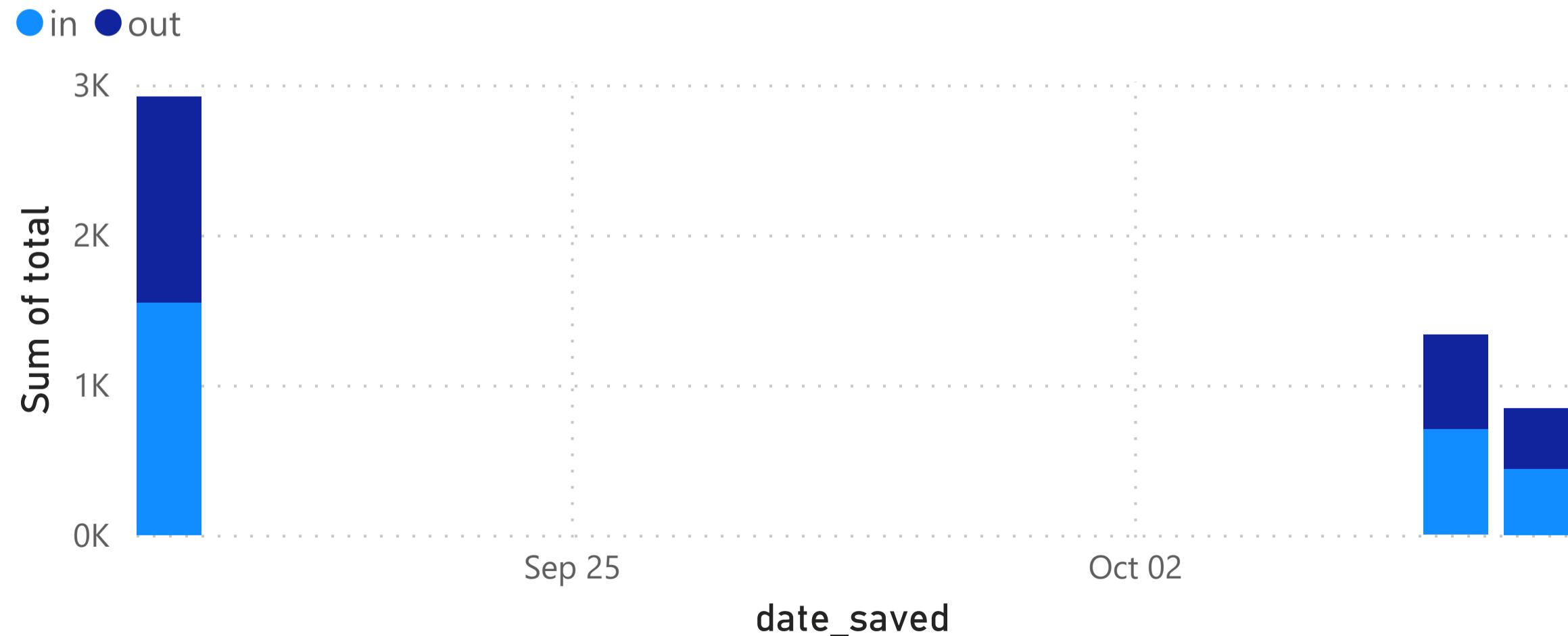
5103

Total vehicles counted

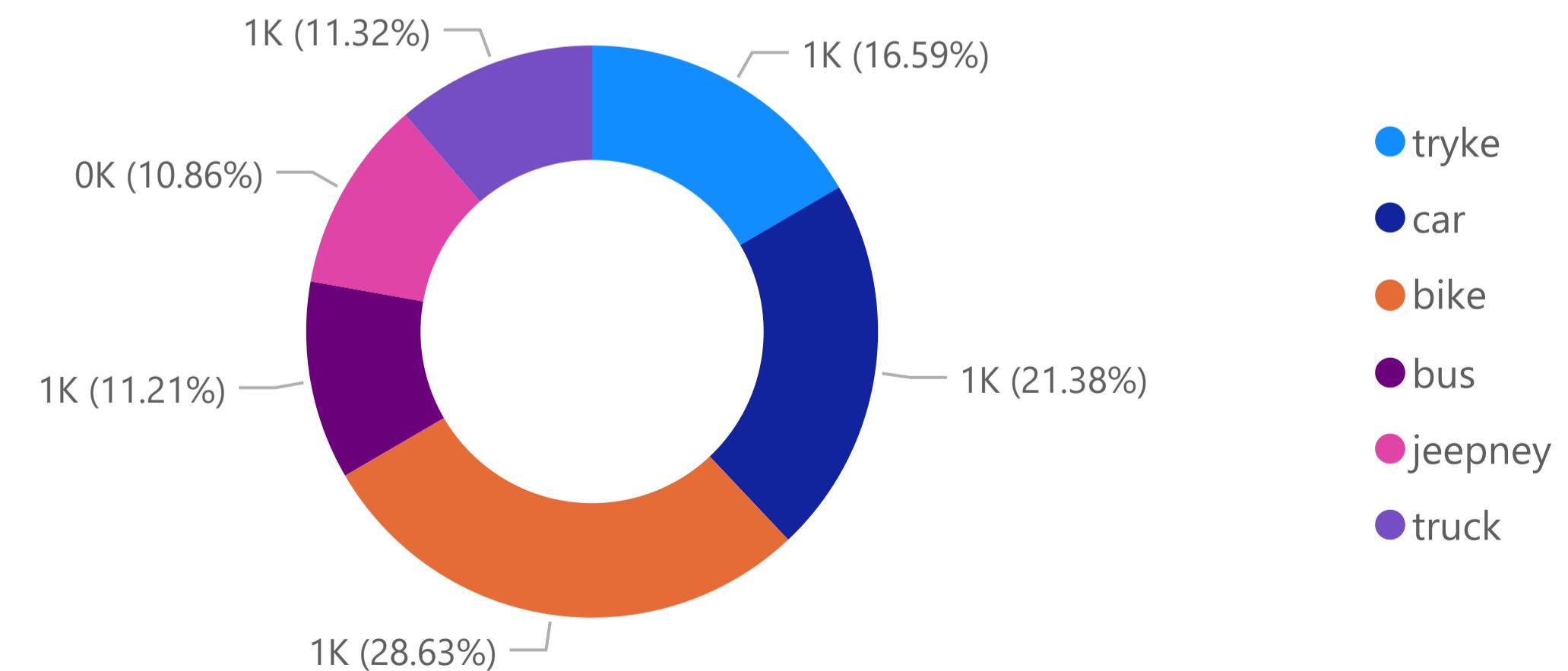
Vehicle count per day



Vehicle actions per day



Vehicle type count for sensor 2





Sensor 3

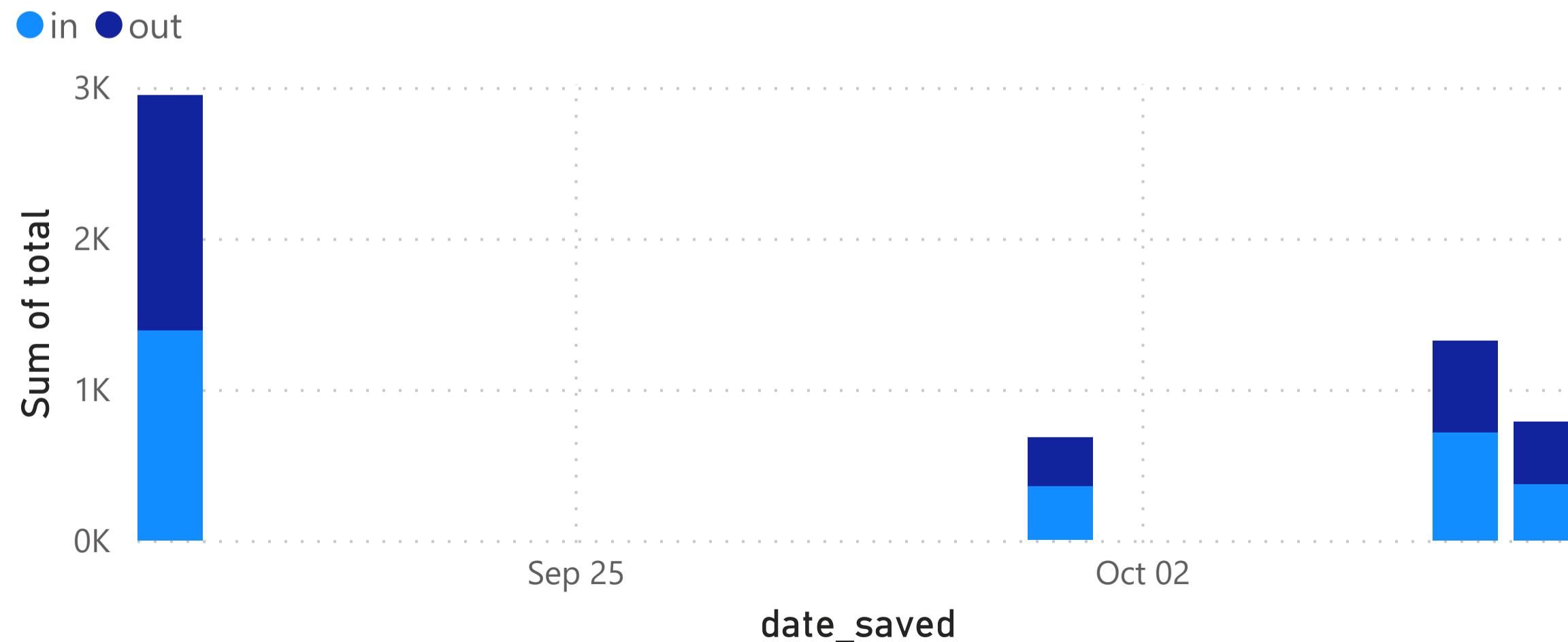
5737

Total vehicles counted

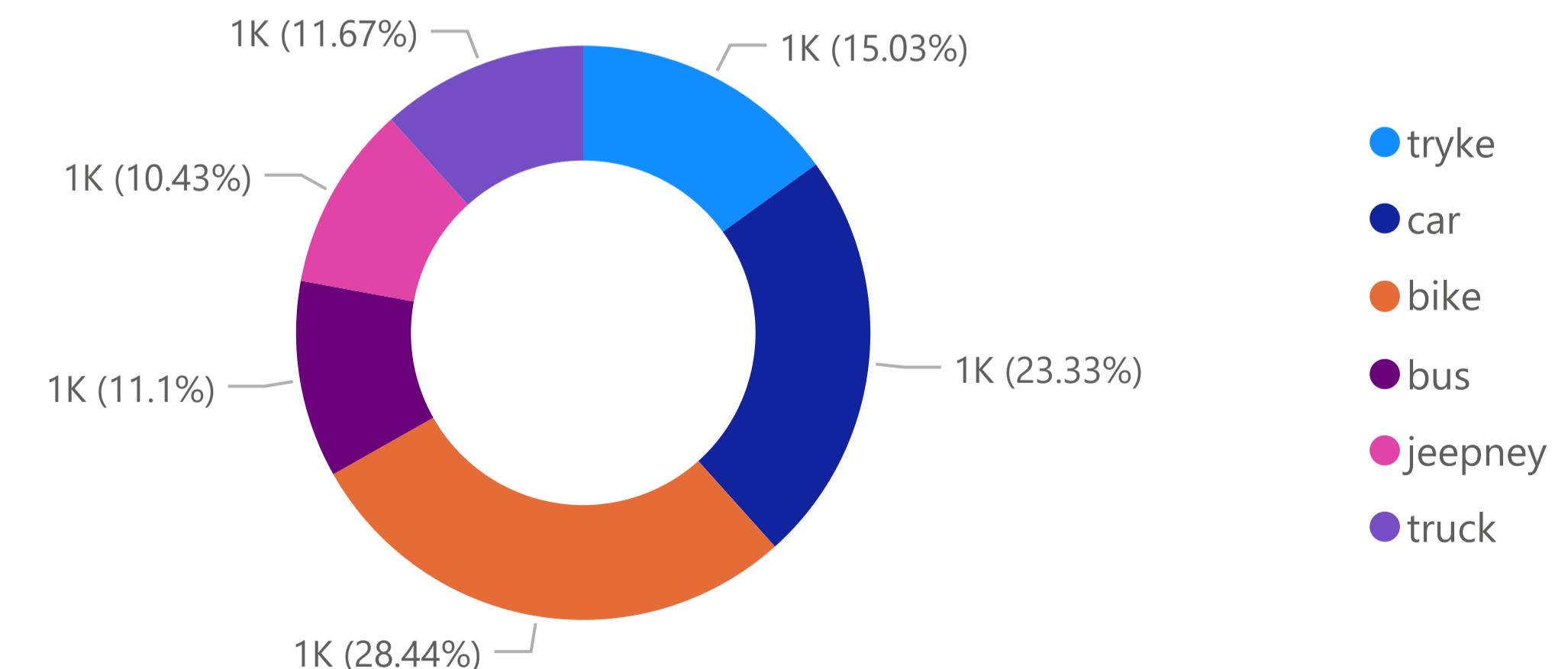
Vehicle count per day



Vehicle actions per day



Vehicle type count for sensor 3





Sensor 4

4764

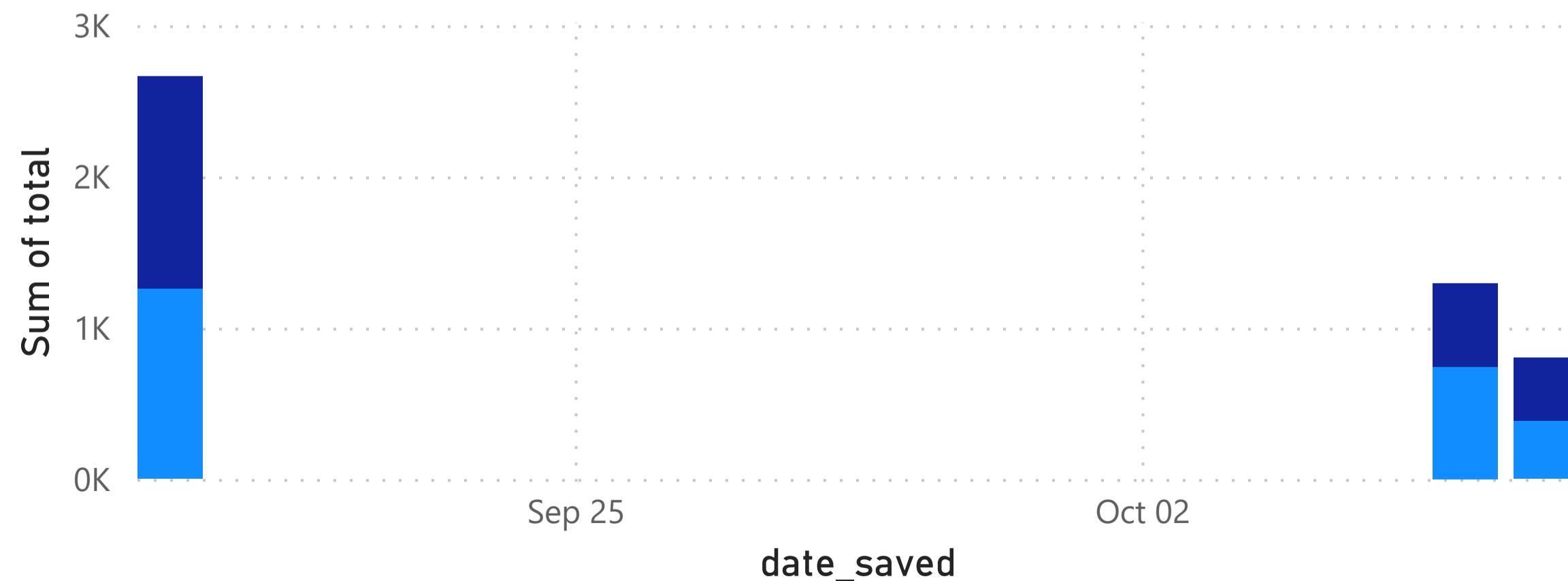
Total vehicles counted

Vehicle count per day

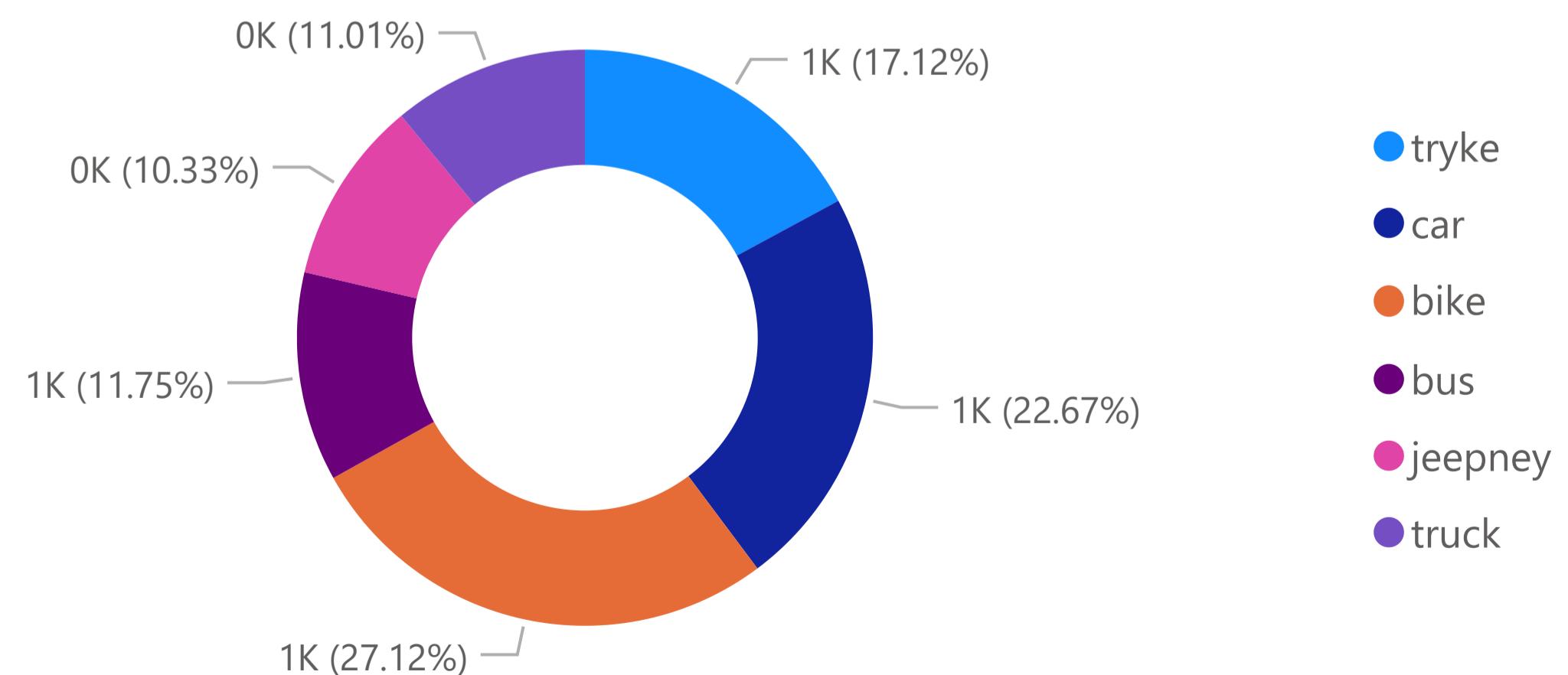


Vehicle actions per day

in out



Vehicle type count for sensor 4





Sensor 5

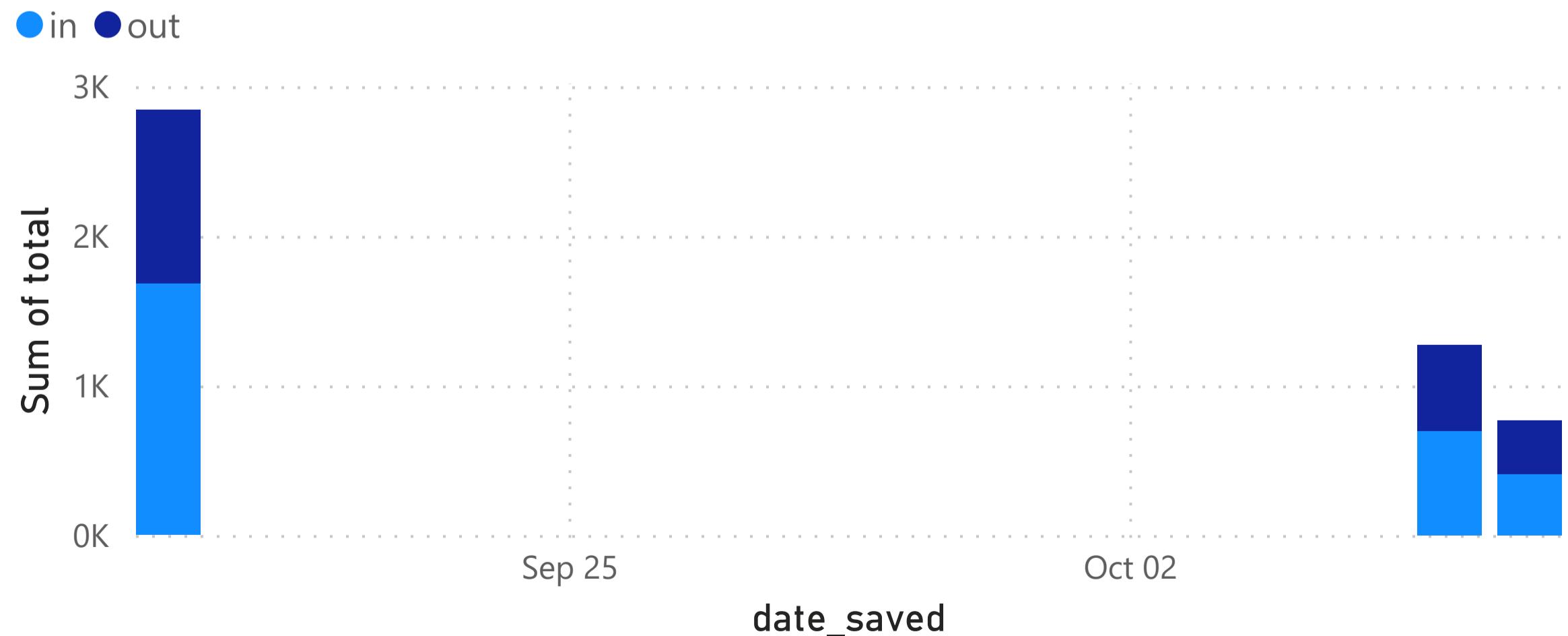
4883

Total vehicles counted

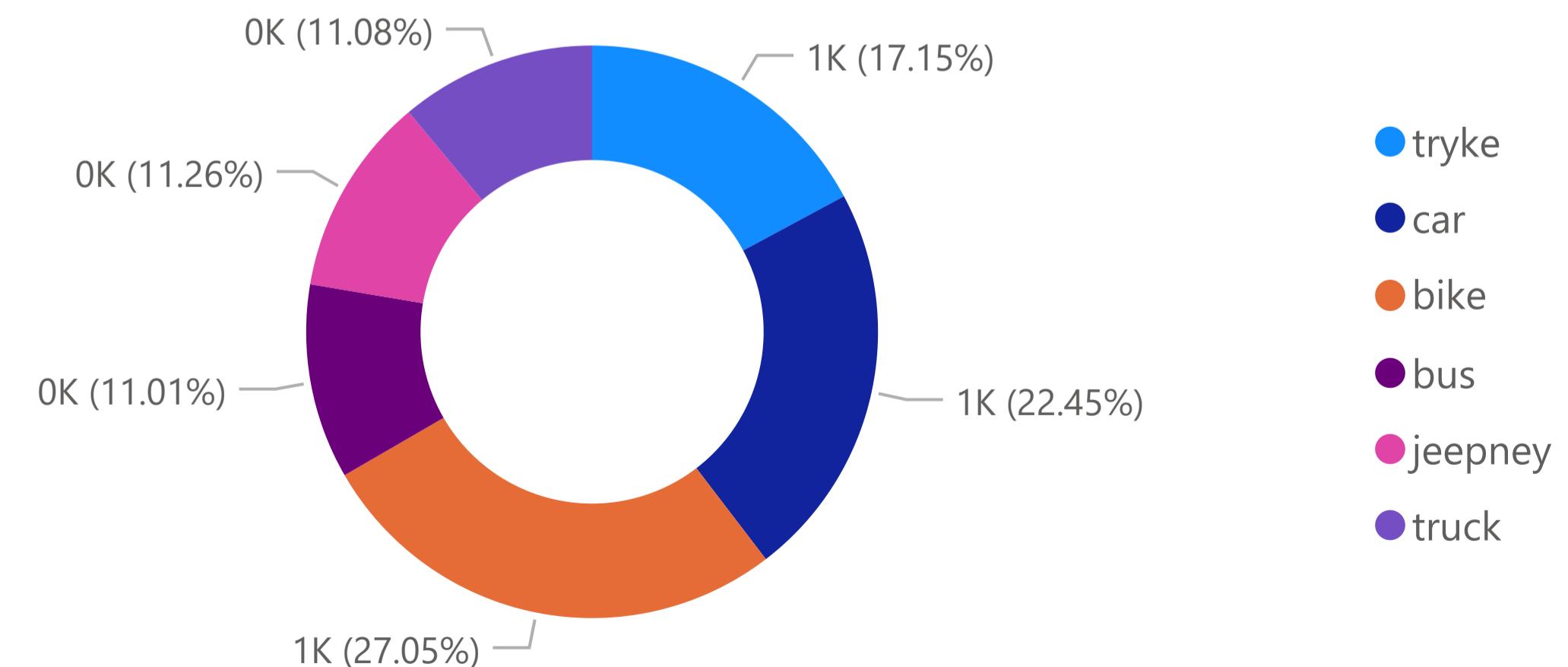
Vehicle count per day



Vehicle actions per day



Vehicle type count for sensor 5





Sensor 6

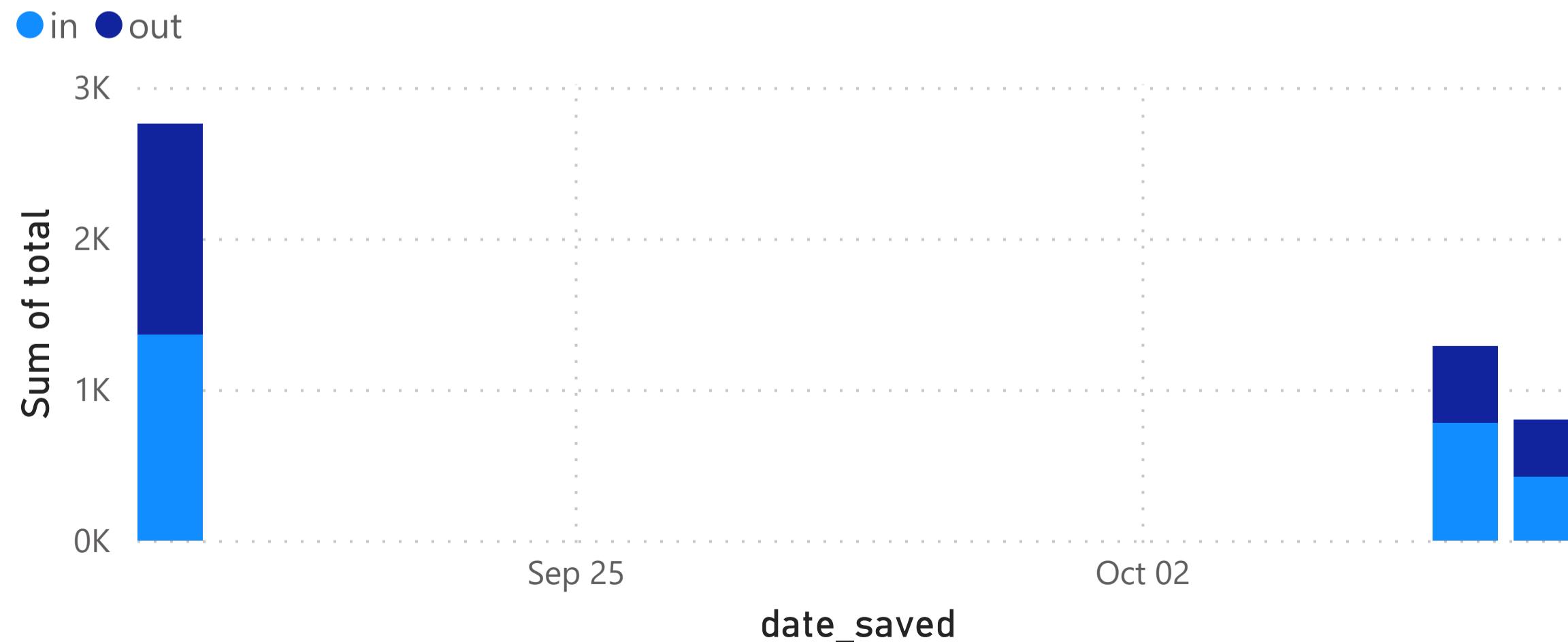
4842

Total vehicles counted

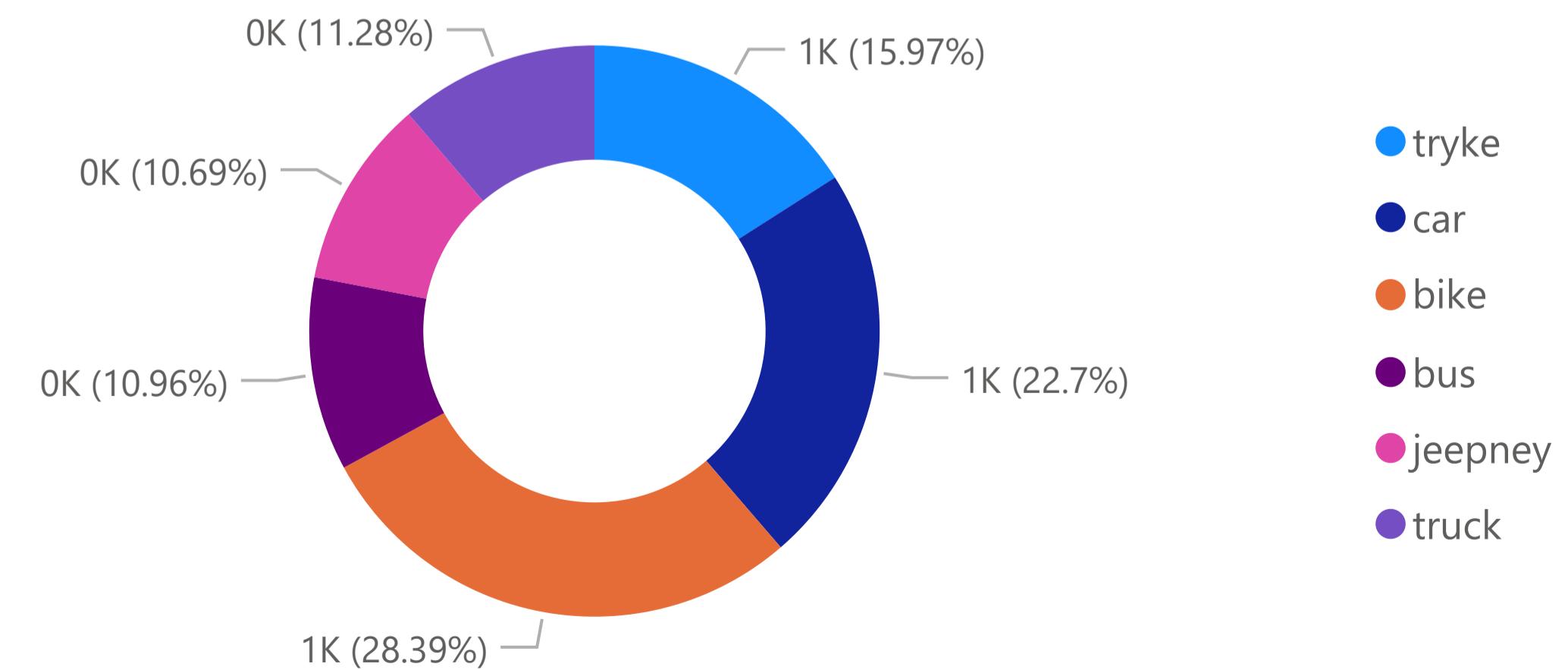
Vehicle count per day



Vehicle actions per day



Vehicle type count for sensor 6





Sensor 7

4468

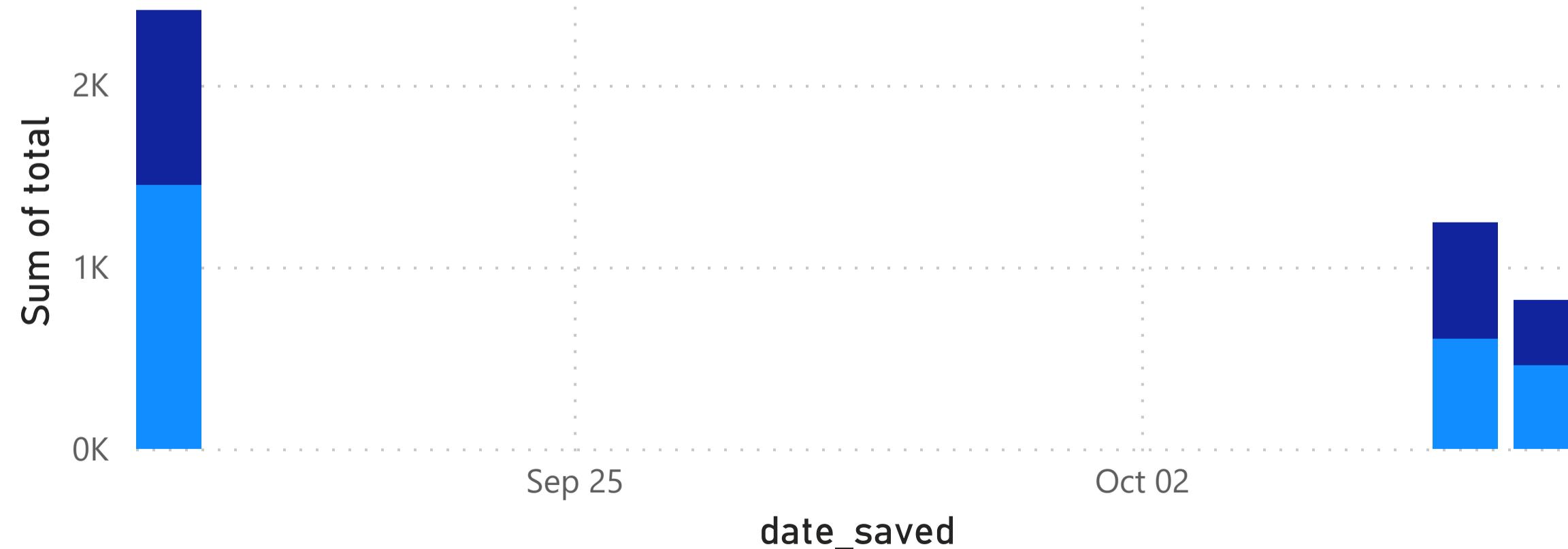
Total vehicles counted

Vehicle count per day

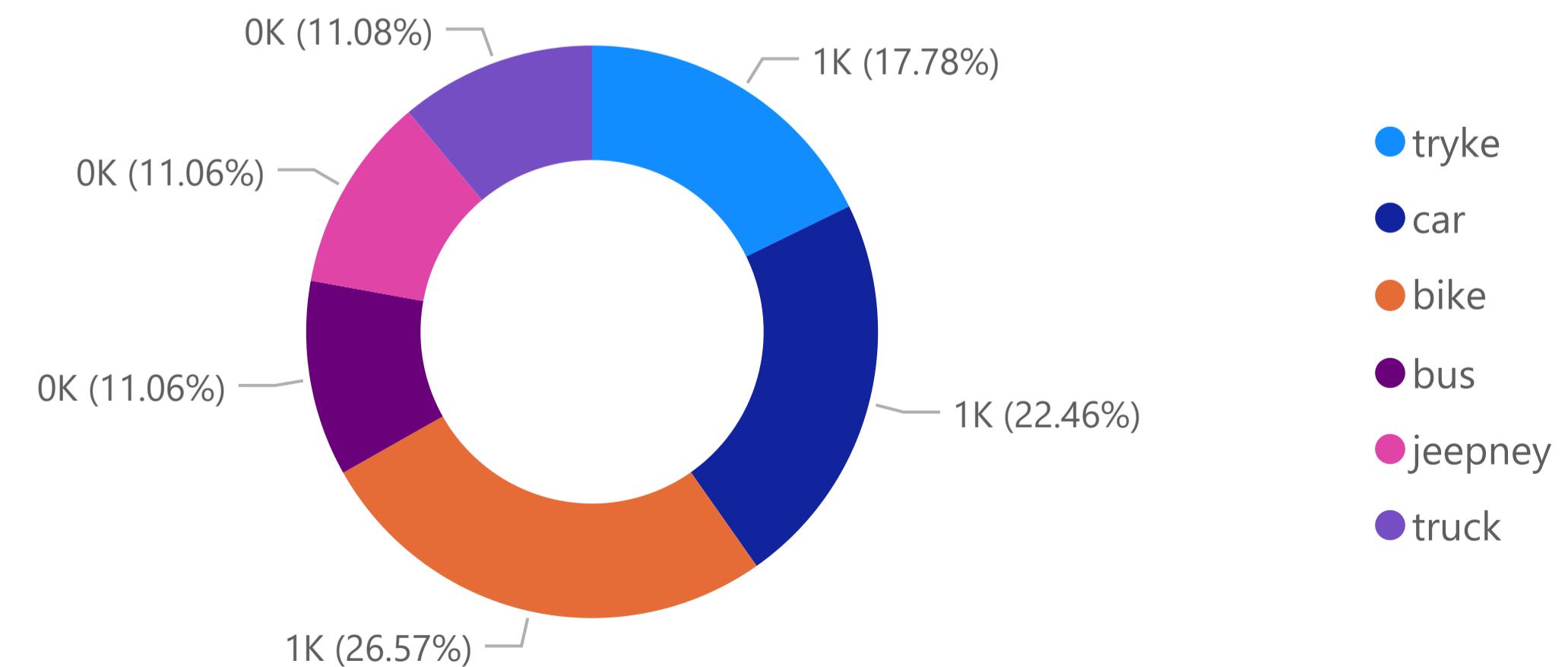


Vehicle actions per day

● in ● out



Vehicle type count for sensor 7





Sensor 8

4458

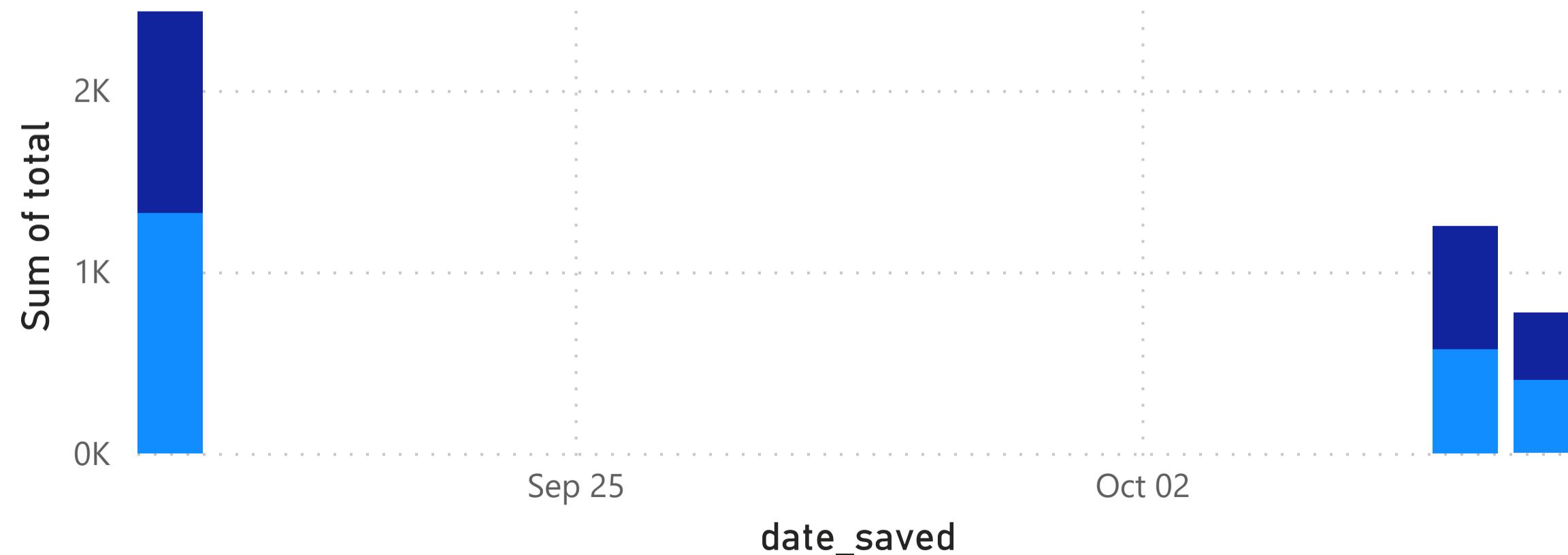
Total vehicles counted

Vehicle count per day

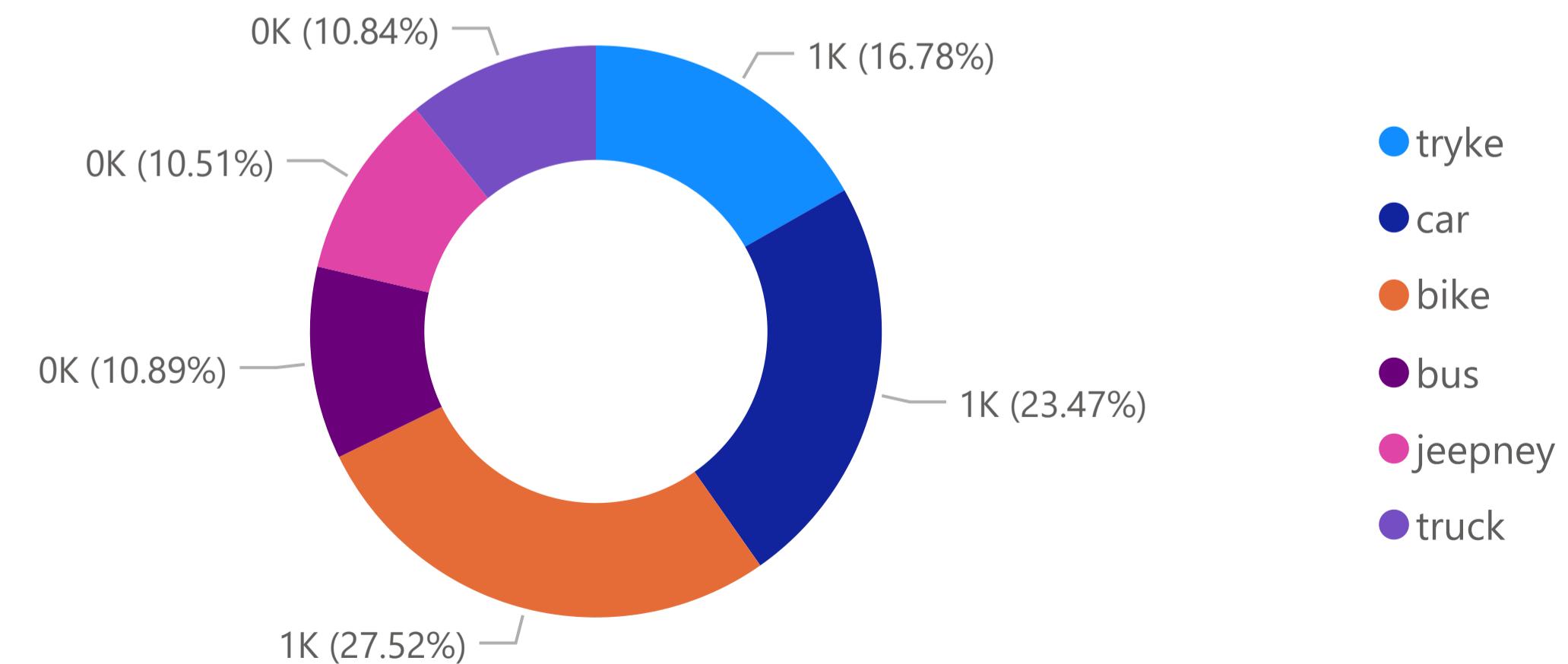


Vehicle actions per day

● in ● out



Vehicle type count for sensor 8



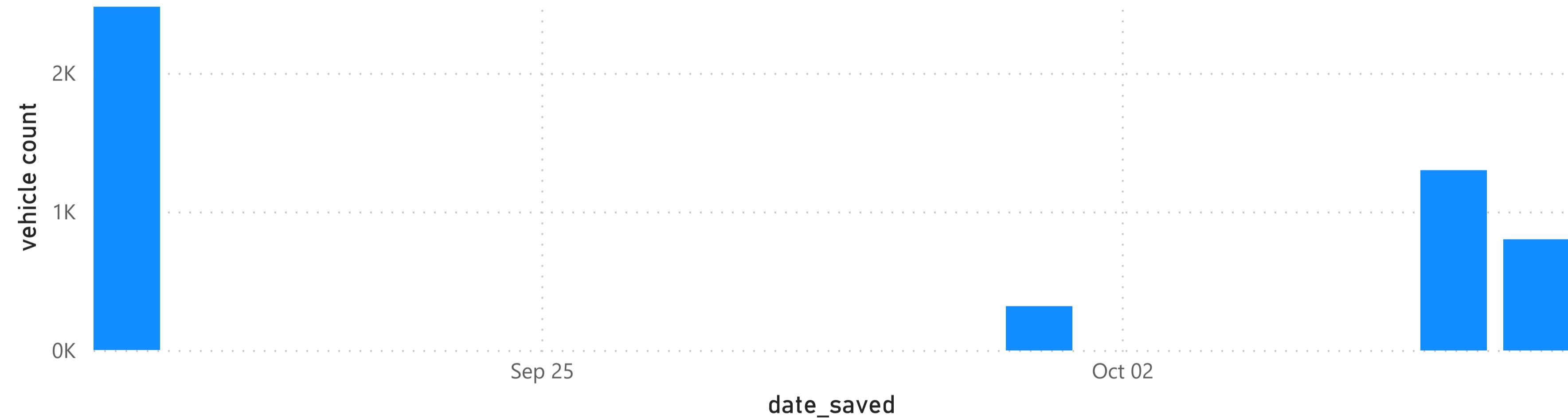


Sensor 9

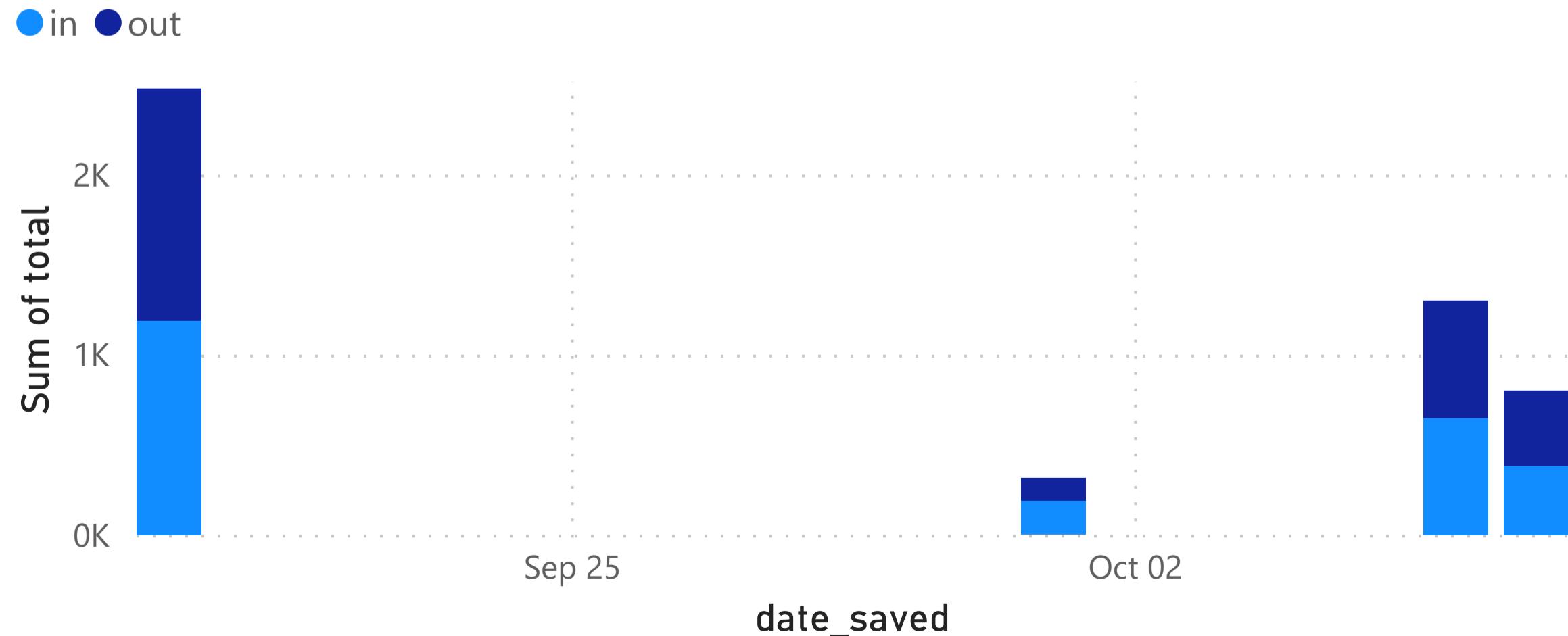
4894

Total vehicles counted

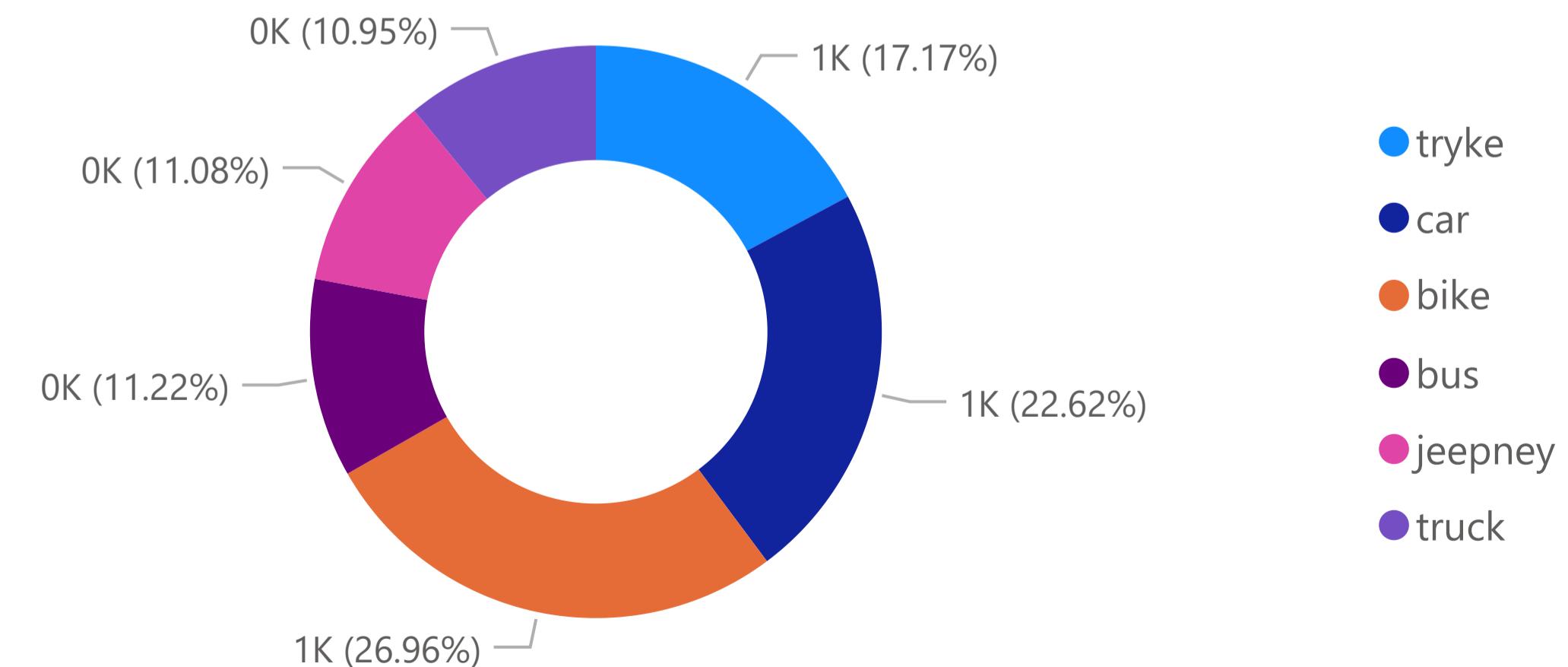
Vehicle count per day



Vehicle actions per day



Vehicle type count for sensor 9



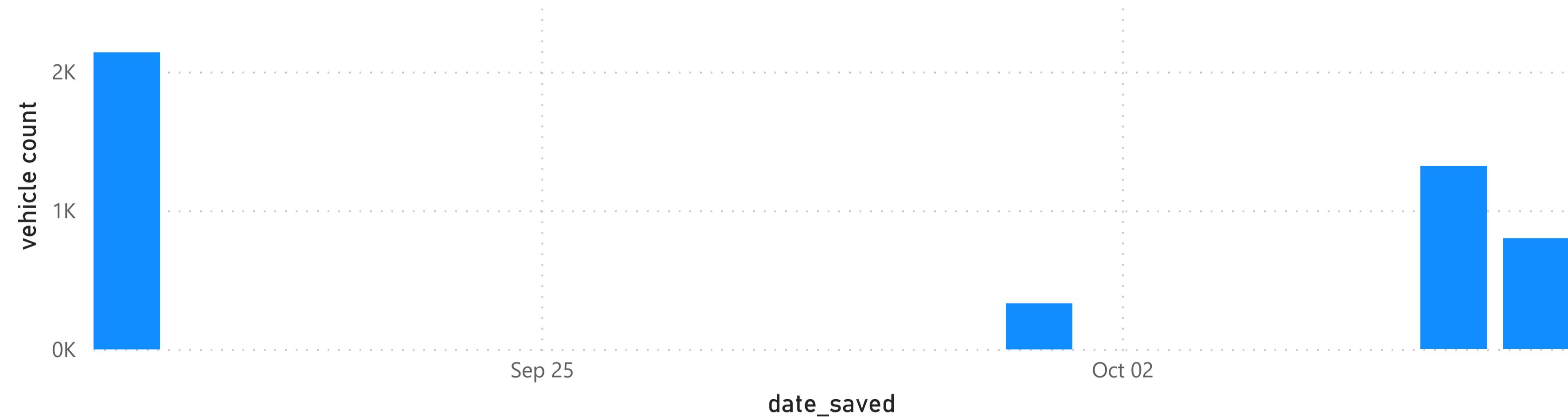


Sensor 10

4590

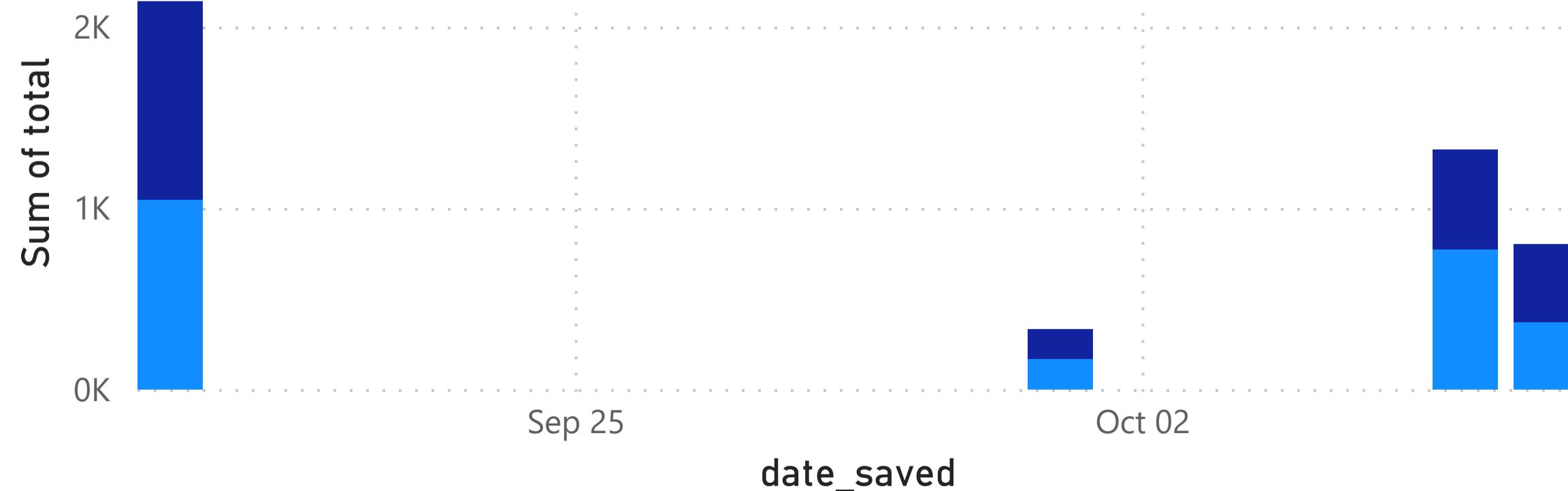
Total vehicles counted

Vehicle count per day



Vehicle actions per day

● in ● out



Vehicle type count for sensor 10

