PART-1

A separate Python script that reads 10 records (of 2 each counter site - ignore the test site) every 5 seconds from traffic counter data file and stores them as separate files (countdata1, countdata2, countdata3, etc.) in the streaming directory on which your application is listening.

```python
import time
from itertools import islice
from itertools import groupby
from itertools import count
inputPath = '/home/bdm/BDM/per-vehicle-records-2021-01-29.csv'
outputPath = '/home/bdm/livestream/'
def split_part(t, data_size):
    d = count()
    for _, s in groupby(t, lambda _: next(d) // data_size):
        yield s
def save_file(data, path):
    with open(path, 'w') as f:
        f.writelines(record + '\n' for record in data)
input = (line.rstrip('\n') for line in open(inputPath))
input = islice(input, 1, None)
input = filter(lambda x: "test" not in x.split(',')[10].lower(), input)
# spiting in part
inputpart =split_part(input, 10)
read = 0
#stores part as separate files (countdata1,countdata2, countdata3, etc.) in the streaming directory
for part in inputpart:
    read += 1
    save_file(part, '{}/countdata{}.csv'.format(outputPath, read))
    time.sleep(5)
```

```
bdm@s1: ~/livestream
countdata251.csv  countdata445.csv  countdata639.csv  countdata832.csv
countdata252.csv  countdata446.csv  countdata63.csv   countdata833.csv
countdata253.csv  countdata447.csv  countdata640.csv  countdata834.csv
countdata254.csv  countdata448.csv  countdata641.csv  countdata835.csv
countdata255.csv  countdata449.csv  countdata642.csv  countdata836.csv
countdata256.csv  countdata44.csv   countdata643.csv  countdata837.csv
countdata257.csv  countdata450.csv  countdata644.csv  countdata838.csv
countdata258.csv  countdata451.csv  countdata645.csv  countdata839.csv
countdata259.csv  countdata452.csv  countdata646.csv  countdata83.csv
countdata25.csv   countdata453.csv  countdata647.csv  countdata840.csv
countdata260.csv  countdata454.csv  countdata648.csv  countdata841.csv
countdata261.csv  countdata455.csv  countdata649.csv  countdata842.csv
countdata262.csv  countdata456.csv  countdata64.csv   countdata843.csv
countdata263.csv  countdata457.csv  countdata650.csv  countdata844.csv
countdata264.csv  countdata458.csv  countdata651.csv  countdata845.csv
countdata265.csv  countdata459.csv  countdata652.csv  countdata846.csv
countdata266.csv  countdata45.csv   countdata653.csv  countdata847.csv
countdata267.csv  countdata460.csv  countdata654.csv  countdata848.csv
countdata268.csv  countdata461.csv  countdata655.csv  countdata849.csv
countdata269.csv  countdata462.csv  countdata656.csv  countdata84.csv
countdata26.csv   countdata463.csv  countdata657.csv  countdata850.csv
countdata270.csv  countdata464.csv  countdata658.csv  countdata851.csv
countdata271.csv  countdata465.csv  countdata659.csv  countdata852.csv
countdata272.csv  countdata466.csv  countdata65.csv   countdata853.csv
countdata273.csv  countdata467.csv  countdata660.csv  countdata854.csv
countdata274.csv  countdata468.csv  countdata661.csv  countdata855.csv
countdata275.csv  countdata469.csv  countdata662.csv  countdata856.csv
countdata276.csv  countdata46.csv   countdata663.csv  countdata857.csv
countdata277.csv  countdata470.csv  countdata664.csv  countdata858.csv
countdata278.csv  countdata471.csv  countdata665.csv  countdata85.csv
countdata279.csv  countdata472.csv  countdata666.csv  countdata86.csv
countdata27.csv   countdata473.csv  countdata667.csv  countdata87.csv
countdata280.csv  countdata474.csv  countdata668.csv  countdata88.csv
countdata281.csv  countdata475.csv  countdata669.csv  countdata89.csv
countdata282.csv  countdata476.csv  countdata66.csv   countdata8.csv
countdata283.csv  countdata477.csv  countdata670.csv  countdata90.csv
countdata284.csv  countdata478.csv  countdata671.csv  countdata91.csv
countdata285.csv  countdata479.csv  countdata672.csv  countdata92.csv
countdata286.csv  countdata47.csv   countdata673.csv  countdata93.csv
countdata287.csv  countdata480.csv  countdata674.csv  countdata94.csv
countdata288.csv  countdata481.csv  countdata675.csv  countdata95.csv
countdata289.csv  countdata482.csv  countdata676.csv  countdata96.csv
countdata28.csv   countdata483.csv  countdata677.csv  countdata97.csv
countdata290.csv  countdata484.csv  countdata678.csv  countdata98.csv
countdata291.csv  countdata485.csv  countdata679.csv  countdata99.csv
countdata292.csv  countdata486.csv  countdata67.csv   countdata9.csv
countdata293.csv  countdata487.csv  countdata680.csv
countdata294.csv  countdata488.csv  countdata681.csv
bdm@s1:~/livestream$
```

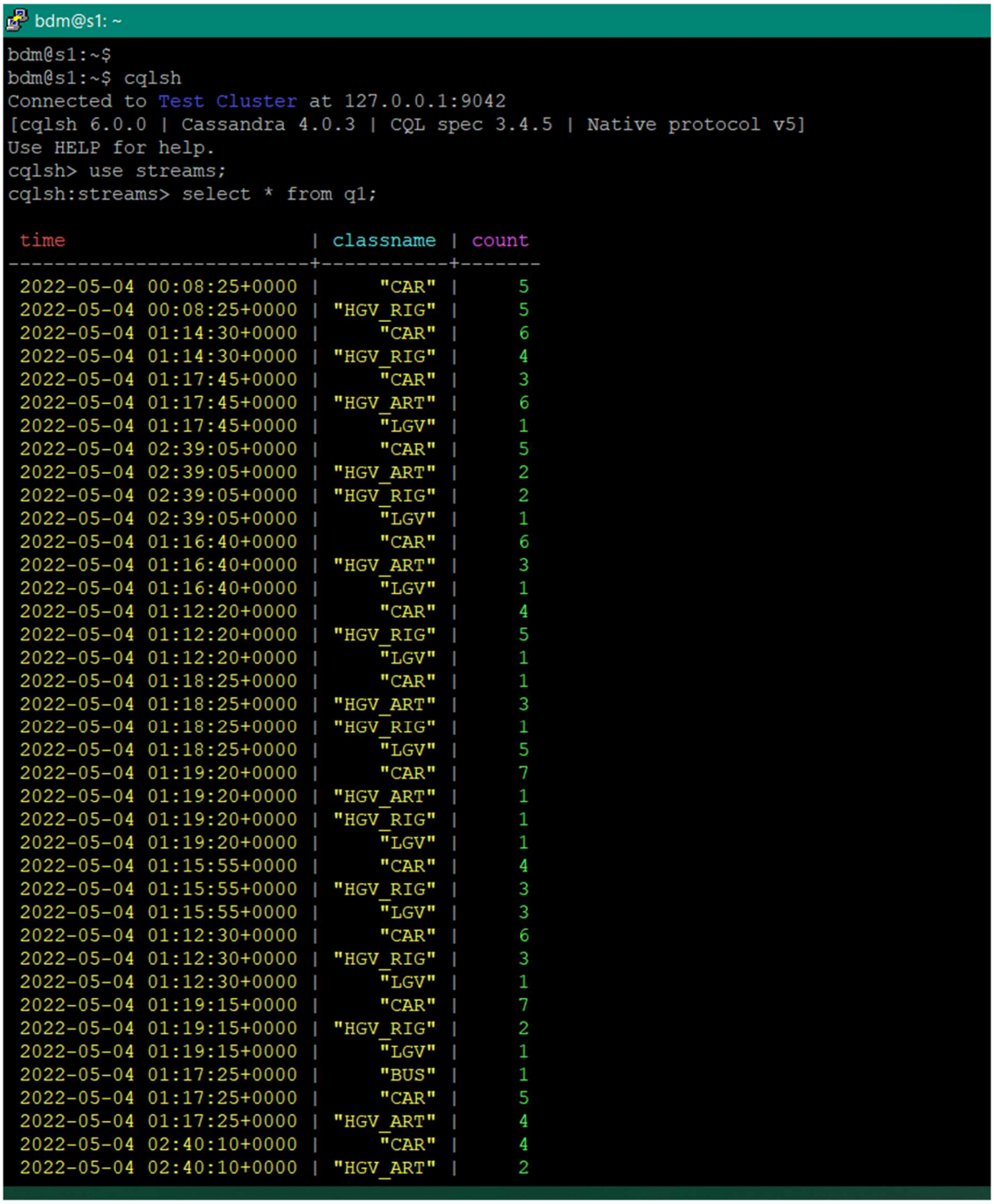
PART 2

Q1. Show total number of counts (on each site of M50) by vehicle class.

```
from pyspark.streaming import StreamingContext
from pyspark.sql import SparkSession
spark=SparkSession.builder.getOrCreate()
from pyspark.sql import Row
import time


def Stream1(time, rdd):
    try:
        df = spark.createDataFrame(rdd.map(\
        lambda row: Row(time=time, classname=row[0], count=row[1])))
        df.write.format("org.apache.spark.sql.cassandra")\
        .options(table="q1", keyspace="streams")\
        .save(mode="append")
    except:
        pass




ssc = StreamingContext(sc, 5)
lines = ssc.textFileStream("/home/bdm/livestream/")
records=lines.map(lambda line:line.split(","))
q1=records.map(lambda x: ((x[14]),1))
q1=q1.reduceByKey(lambda a,b : a+b)
q1.foreachRDD(Stream1)
q1.pprint()

ssc.start()
time.sleep(100)
ssc.stop(stopSparkContext=False)
```

```
-------------------------------------------
Time: 2022-05-04 02:38:55
-------------------------------------------
('"CAR"', 6)
('"HGV_ART"', 2)
('"LGV"', 1)
('"CARAVAN"', 1)

-------------------------------------------
Time: 2022-05-04 02:39:00
-------------------------------------------
('"HGV_ART"', 2)
('"HGV_RIG"', 2)
('"CARAVAN"', 1)
('"CAR"', 4)
('"LGV"', 1)

-------------------------------------------
Time: 2022-05-04 02:39:05
-------------------------------------------
```

Q1 Cassandra Output



Q2. Compute the average speed (on each site on M50) by vehicle class.

```
from pyspark.streaming import StreamingContext
from pyspark.sql import SparkSession
spark=SparkSession.builder.getOrCreate()
from pyspark.sql import Row
import time


def Stream2(time, rdd):
    try:
        df = spark.createDataFrame(rdd.map(\
        lambda row: Row(time=time, classname=row[0],avg_speed=row[1],)))
        df.write.format("org.apache.spark.sql.cassandra")\
        .options(table="q2", keyspace="streams")\
        .save(mode="append")
    except:
        pass




ssc = StreamingContext(sc, 5)
lines = ssc.textFileStream("/home/bdm/livestream/")
records=lines.map(lambda line:line.split(","))
q2=records.map(lambda x: (x[14].strip('"'), float(x[18].strip('"')))).groupByKey().map(lambda x: (x[0], list(x[1]))).map(lambda x
q2.foreachRDD(Stream2)
q2.pprint()

ssc.start()
time.sleep(100)
ssc.stop(stopSparkContext=False)
```

```
Time: 2022-05-04 02:40:35
-------------------------------------------
('HGV_ART', 92.5)
('CARAVAN', 87.0)
('LGV', 96.0)
('CAR', 119.0)

-------------------------------------------
Time: 2022-05-04 02:40:40
-------------------------------------------
('HGV_ART', 88.77777777777777)
('CAR', 84.0)

-------------------------------------------
Time: 2022-05-04 02:40:45
-------------------------------------------
('HGV_ART', 86.0)
('LGV', 125.33333333333333)
('CAR', 110.0)
```

Q2 Cassandra Output

```
bdm@s1: ~
cqlsh:streams>
cqlsh:streams> select * from q2;

 time                     | classname | avg_speed
--------------------------+-----------+-----------
 2022-05-04 02:35:50+0000 |       CAR |        91
 2022-05-04 02:35:50+0000 |   HGV_ART |      86.6
 2022-05-04 02:35:50+0000 |       LGV |  94.66667
 2022-05-04 02:35:15+0000 |       CAR |    101.75
 2022-05-04 02:35:15+0000 |   HGV_ART |        91
 2022-05-04 02:35:15+0000 |       LGV |       102
 2022-05-04 02:22:20+0000 |       CAR | 118.34444
 2022-05-04 02:22:20+0000 |       LGV |     127.9
 2022-05-04 02:41:05+0000 |       BUS |        95
 2022-05-04 02:41:05+0000 |       CAR |       113
 2022-05-04 02:41:05+0000 |   HGV_ART |  75.33333
 2022-05-04 02:41:05+0000 |   HGV_RIG |        79
 2022-05-04 02:41:05+0000 |       LGV |        83
 2022-05-04 02:23:25+0000 |       CAR |    117.95
 2022-05-04 02:23:25+0000 |   HGV_RIG |    96.025
 2022-05-04 02:23:25+0000 |       LGV |    114.45
 2022-05-04 02:40:45+0000 |       CAR |       110
 2022-05-04 02:40:45+0000 |   HGV_ART |        86
 2022-05-04 02:40:45+0000 |       LGV | 125.33333
 2022-05-04 02:23:05+0000 |       CAR |     115.3
 2022-05-04 02:23:05+0000 |       LGV |    110.95
 2022-05-04 02:23:05+0000 |     MBIKE |     129.5
 2022-05-04 02:35:40+0000 |       CAR |        93
 2022-05-04 02:35:40+0000 |   HGV_ART |  81.16667
 2022-05-04 02:35:40+0000 |       LGV |      94.5
 2022-05-04 02:35:05+0000 |       CAR |  95.66667
 2022-05-04 02:35:05+0000 |   HGV_ART |      89.5
 2022-05-04 02:35:05+0000 |       LGV |       109
 2022-05-04 02:35:25+0000 |       CAR |        92
 2022-05-04 02:35:25+0000 |   HGV_ART |        90
 2022-05-04 02:35:25+0000 |   HGV_RIG |        94
 2022-05-04 02:35:25+0000 |       LGV |     103.8
 2022-05-04 02:41:40+0000 |       CAR |  96.66667
 2022-05-04 02:41:40+0000 |   HGV_ART |        86
 2022-05-04 02:41:40+0000 |       LGV |       126
 2022-05-04 02:41:25+0000 |       CAR |  81.66667
 2022-05-04 02:41:25+0000 |   HGV_ART |        73
 2022-05-04 02:41:25+0000 |   HGV_RIG |        24
 2022-05-04 02:22:00+0000 |       CAR |    109.78
 2022-05-04 02:22:00+0000 |   HGV_RIG |       117
 2022-05-04 02:22:00+0000 |       LGV |    108.25
 2022-05-04 02:21:50+0000 |       CAR | 115.18571
 2022-05-04 02:21:50+0000 |   HGV_RIG |        87
 2022-05-04 02:21:50+0000 |       LGV |     110.4
```

Q3. Find the top 3 busiest counter sites on M50.

```
from pyspark.streaming import StreamingContext
from pyspark.sql import SparkSession
spark=SparkSession.builder.getOrCreate()
from pyspark.sql import Row
import time


def Stream3(time, rdd):
    try:
        df = spark.createDataFrame(rdd.map(\
        lambda row: Row(time=time, site=row[0], count=row[1])))
        df.write.format("org.apache.spark.sql.cassandra")\
        .options(table="q3", keyspace="streams")\
        .save(mode="append")
    except:
        pass

ssc = StreamingContext(sc, 5)
lines = ssc.textFileStream("/home/bdm/livestream/")
records=lines.map(lambda line:line.split(","))
q3=records.map(lambda x: ((x[0]),1))
q3=q3.reduceByKey(lambda x,y: x+y )
q3=q3.transform(
    lambda rdd:rdd.sortBy(lambda x:x[1],ascending=False)
)
q3.foreachRDD(Stream3)
q3.pprint()

ssc.start()
time.sleep(300)
ssc.stop(stopSparkContext=False)
```

```
-------------------------------------------
Time: 2022-05-04 02:42:15
-------------------------------------------
('"000000020076"', 7)
('"000000020077"', 3)

-------------------------------------------
Time: 2022-05-04 02:42:20
-------------------------------------------
('"000000020077"', 9)
('"000000020078"', 1)

-------------------------------------------
Time: 2022-05-04 02:42:25
-------------------------------------------
('"000000020078"', 8)
('"000000020079"', 2)

-------------------------------------------
Time: 2022-05-04 02:42:30
```

Q3 Cassandra Output

Q4. Find total number of counts for HGVs on M50.

```
from pyspark.streaming import StreamingContext
from pyspark.sql import SparkSession
spark=SparkSession.builder.getOrCreate()
from pyspark.sql import Row
import time


def Stream4(time, rdd):
    try:
        df = spark.createDataFrame(rdd.map(\
        lambda row: Row(time=time, m50parts=row[0], count=row[1])))
        df.write.format("org.apache.spark.sql.cassandra")\
        .options(table="q4", keyspace="streams")\
        .save(mode="append")
    except:
        pass

ssc = StreamingContext(sc, 5)
lines = ssc.textFileStream("/home/bdm/livestream/")
records=lines.map(lambda line:line.split(","))
q4=records.filter(lambda x: "HGV" in x[14])
q4=records.map(lambda x: (x[10].strip('"'), 1))
q4=q4.reduceByKey(lambda x, y: x + y)
q4.foreachRDD(Stream4)
q4.pprint()

ssc.start()
time.sleep(100)
ssc.stop(stopSparkContext=False)
```

```
-------------------------------------------
Time: 2022-05-04 02:47:20
-------------------------------------------
('Northbound 2 (fast)', 4)
('Southbound 1 (slow)', 3)
('Southbound 2 (fast)', 3)

-------------------------------------------
Time: 2022-05-04 02:47:25
-------------------------------------------
('Southbound 2 (fast)', 2)
('Southbound 1 (slow)', 3)
('Northbound 1 (slow)', 4)
('Northbound 2 (fast)', 1)

-------------------------------------------
Time: 2022-05-04 02:47:30
-------------------------------------------
('Southbound 2 (fast)', 2)
('Southbound 1 (slow)', 3)
```

Q4 Cassandra Output