

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.

```
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@sl: ~/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
countdata270.csv countdata463.csv countdata657.csv countdata850.csv
countdata271.csv countdata464.csv countdata658.csv countdata851.csv
countdata272.csv countdata465.csv countdata659.csv countdata852.csv
countdata273.csv countdata466.csv countdata65.csv countdata853.csv
countdata274.csv countdata467.csv countdata660.csv countdata854.csv
countdata275.csv countdata468.csv countdata661.csv countdata855.csv
countdata276.csv countdata469.csv countdata662.csv countdata856.csv
countdata277.csv countdata470.csv countdata663.csv countdata857.csv
countdata278.csv countdata471.csv countdata664.csv countdata858.csv
countdata279.csv countdata472.csv countdata665.csv countdata85.csv
countdata27.csv countdata473.csv countdata666.csv countdata86.csv
countdata280.csv countdata474.csv countdata667.csv countdata87.csv
countdata281.csv countdata475.csv countdata668.csv countdata88.csv
countdata282.csv countdata476.csv countdata669.csv countdata89.csv
countdata283.csv countdata477.csv countdata66.csv countdata8.csv
countdata284.csv countdata478.csv countdata670.csv countdata90.csv
countdata285.csv countdata479.csv countdata671.csv countdata91.csv
countdata286.csv countdata47.csv countdata672.csv countdata92.csv
countdata287.csv countdata480.csv countdata673.csv countdata93.csv
countdata288.csv countdata481.csv countdata674.csv countdata94.csv
countdata289.csv countdata482.csv countdata675.csv countdata95.csv
countdata28.csv countdata483.csv countdata676.csv countdata96.csv
countdata290.csv countdata484.csv countdata677.csv countdata97.csv
countdata291.csv countdata485.csv countdata678.csv countdata98.csv
countdata292.csv countdata486.csv countdata679.csv countdata99.csv
countdata293.csv countdata487.csv countdata67.csv countdata9.csv
countdata294.csv countdata488.csv countdata680.csv
countdata488.csv countdata681.csv
```

bdm@sl:~/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

```
bdm@s1: ~  
bdm@s1:~$  
bdm@s1:~$ cqlsh  
Connected to Test Cluster at 127.0.0.1:9042  
[cqlsh 6.0.0 | Cassandra 4.0.3 | CQL spec 3.4.5 | Native protocol v5]  
Use HELP for help.  
cqlsh> use streams;  
cqlsh:streams> select * from q1;
```

time	classname	count
2022-05-04 00:08:25+0000	"CAR"	5
2022-05-04 00:08:25+0000	"HGV_RIG"	5
2022-05-04 01:14:30+0000	"CAR"	6
2022-05-04 01:14:30+0000	"HGV_RIG"	4
2022-05-04 01:17:45+0000	"CAR"	3
2022-05-04 01:17:45+0000	"HGV_ART"	6
2022-05-04 01:17:45+0000	"LGV"	1
2022-05-04 02:39:05+0000	"CAR"	5
2022-05-04 02:39:05+0000	"HGV_ART"	2
2022-05-04 02:39:05+0000	"HGV_RIG"	2
2022-05-04 02:39:05+0000	"LGV"	1
2022-05-04 01:16:40+0000	"CAR"	6
2022-05-04 01:16:40+0000	"HGV_ART"	3
2022-05-04 01:16:40+0000	"LGV"	1
2022-05-04 01:12:20+0000	"CAR"	4
2022-05-04 01:12:20+0000	"HGV_RIG"	5
2022-05-04 01:12:20+0000	"LGV"	1
2022-05-04 01:18:25+0000	"CAR"	1
2022-05-04 01:18:25+0000	"HGV_ART"	3
2022-05-04 01:18:25+0000	"HGV_RIG"	1
2022-05-04 01:18:25+0000	"LGV"	5
2022-05-04 01:19:20+0000	"CAR"	7
2022-05-04 01:19:20+0000	"HGV_ART"	1
2022-05-04 01:19:20+0000	"HGV_RIG"	1
2022-05-04 01:19:20+0000	"LGV"	1
2022-05-04 01:15:55+0000	"CAR"	4
2022-05-04 01:15:55+0000	"HGV_RIG"	3
2022-05-04 01:15:55+0000	"LGV"	3
2022-05-04 01:12:30+0000	"CAR"	6
2022-05-04 01:12:30+0000	"HGV_RIG"	3
2022-05-04 01:12:30+0000	"LGV"	1
2022-05-04 01:19:15+0000	"CAR"	7
2022-05-04 01:19:15+0000	"HGV_RIG"	2
2022-05-04 01:19:15+0000	"LGV"	1
2022-05-04 01:17:25+0000	"BUS"	1
2022-05-04 01:17:25+0000	"CAR"	5
2022-05-04 01:17:25+0000	"HGV_ART"	4
2022-05-04 02:40:10+0000	"CAR"	4
2022-05-04 02:40:10+0000	"HGV_ART"	2

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])))
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

```
bdm@s1: ~  
(199 rows)  
cqlsh:streams> select * from q3;
```

time	site	count
2022-05-04 02:43:55+0000	"0000000020243"	3
2022-05-04 02:43:55+0000	"0000000020251"	1
2022-05-04 02:43:55+0000	"0000000020253"	1
2022-05-04 02:43:55+0000	"0000000020254"	5
2022-05-04 02:43:20+0000	"0000000020116"	1
2022-05-04 02:43:20+0000	"0000000020118"	7
2022-05-04 02:43:20+0000	"0000000020151"	2
2022-05-04 02:45:40+0000	"0000000010111"	1
2022-05-04 02:45:40+0000	"000000001012"	9
2022-05-04 01:42:40+0000	"000000001502"	10
2022-05-04 01:43:20+0000	"000000001503"	10
2022-05-04 02:42:55+0000	"0000000020088"	7
2022-05-04 02:42:55+0000	"0000000020089"	3
2022-05-04 01:42:45+0000	"000000001502"	10
2022-05-04 02:44:40+0000	"000000200713"	5
2022-05-04 02:44:40+0000	"000000200714"	3
2022-05-04 02:44:40+0000	"000000200715"	2
2022-05-04 01:41:55+0000	"000000001500"	10
2022-05-04 02:46:20+0000	"000000001014"	7
2022-05-04 02:46:20+0000	"000000001015"	3
2022-05-04 02:46:05+0000	"000000001014"	10
2022-05-04 01:44:30+0000	"000000001505"	10
2022-05-04 01:43:55+0000	"000000001504"	10
2022-05-04 02:45:50+0000	"000000001012"	5
2022-05-04 02:45:50+0000	"000000001013"	5
2022-05-04 02:45:15+0000	"000000201082"	3
2022-05-04 02:45:15+0000	"000000201321"	2
2022-05-04 02:45:15+0000	"000000202103"	1
2022-05-04 02:45:15+0000	"000000202104"	2
2022-05-04 02:45:15+0000	"000000202639"	2
2022-05-04 02:46:40+0000	"000000001015"	10
2022-05-04 02:42:50+0000	"0000000020087"	10
2022-05-04 02:42:15+0000	"0000000020076"	7
2022-05-04 02:42:15+0000	"0000000020077"	3
2022-05-04 01:44:45+0000	"000000001506"	10
2022-05-04 01:45:10+0000	"000000001507"	10
2022-05-04 01:44:40+0000	"000000001506"	10
2022-05-04 01:44:50+0000	"000000001506"	10
2022-05-04 01:46:30+0000	"000000001509"	10
2022-05-04 02:46:50+0000	"000000001015"	10
2022-05-04 01:42:30+0000	"000000001502"	10
2022-05-04 02:45:25+0000	"000000000998"	10
2022-05-04 02:43:40+0000	"0000000020203"	3
2022-05-04 02:43:40+0000	"0000000020204"	2

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

```
bdm@s1: ~  
(211 rows)  
cqlsh:streams> select * from q4;
```

time	m50parts	count
2022-05-04 02:14:35+0000	Northbound 1	7
2022-05-04 02:14:35+0000	Southbound 1	2
2022-05-04 02:14:35+0000	Southbound 2	1
2022-05-04 02:48:20+0000	Northbound 1	1
2022-05-04 02:48:20+0000	Northbound 2	1
2022-05-04 02:48:20+0000	Northbound Off Slip	3
2022-05-04 02:48:20+0000	Southbound 1	2
2022-05-04 02:48:20+0000	Southbound 2	2
2022-05-04 02:48:20+0000	Southbound On Slip	1
2022-05-04 02:47:40+0000	Northbound 1 (slow)	3
2022-05-04 02:47:40+0000	Northbound 2 (fast)	1
2022-05-04 02:47:40+0000	Southbound 1 (slow)	2
2022-05-04 02:47:40+0000	Southbound 2 (fast)	4
2022-05-04 02:14:40+0000	Northbound 1	2
2022-05-04 02:14:40+0000	Southbound 1	2
2022-05-04 02:14:40+0000	Southbound 2	6
2022-05-04 02:48:15+0000	Northbound 1	2
2022-05-04 02:48:15+0000	Northbound Off Slip	4
2022-05-04 02:48:15+0000	Southbound 1	2
2022-05-04 02:48:15+0000	Southbound 2	2
2022-05-04 02:48:50+0000	Northbound	6
2022-05-04 02:48:50+0000	Northbound	1
2022-05-04 02:48:50+0000	Southbound	1
2022-05-04 02:48:50+0000	Southbound	2
2022-05-04 02:14:30+0000	Northbound 1	1
2022-05-04 02:14:30+0000	Northbound 3	1
2022-05-04 02:14:30+0000	Southbound 1	3
2022-05-04 02:14:30+0000	Southbound 2	5
2022-05-04 02:47:25+0000	Northbound 1 (slow)	4
2022-05-04 02:47:25+0000	Northbound 2 (fast)	1
2022-05-04 02:47:25+0000	Southbound 1 (slow)	3
2022-05-04 02:47:25+0000	Southbound 2 (fast)	2
2022-05-04 02:47:30+0000	Northbound 1 (slow)	3
2022-05-04 02:47:30+0000	Northbound 2 (fast)	2
2022-05-04 02:47:30+0000	Southbound 1 (slow)	3
2022-05-04 02:47:30+0000	Southbound 2 (fast)	2
2022-05-04 02:48:05+0000	Northbound	8
2022-05-04 02:48:05+0000	Southbound	2
2022-05-04 02:14:55+0000	Northbound 1	5
2022-05-04 02:14:55+0000	Southbound 1	4
2022-05-04 02:14:55+0000	Southbound 2	1
2022-05-04 02:48:30+0000	Northbound 1	3
2022-05-04 02:48:30+0000	Northbound 2	1
2022-05-04 02:48:30+0000	Southbound 1	4