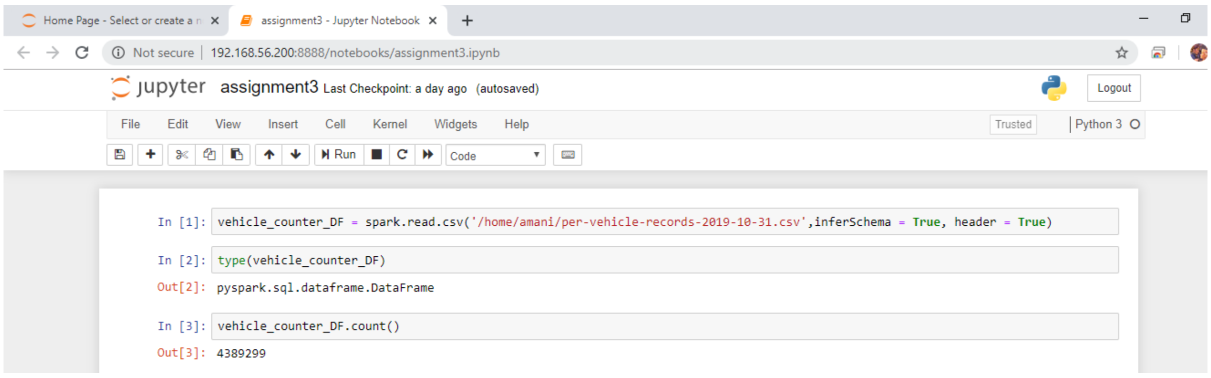
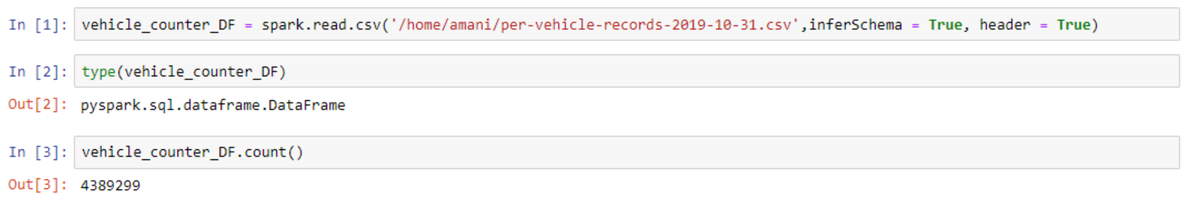
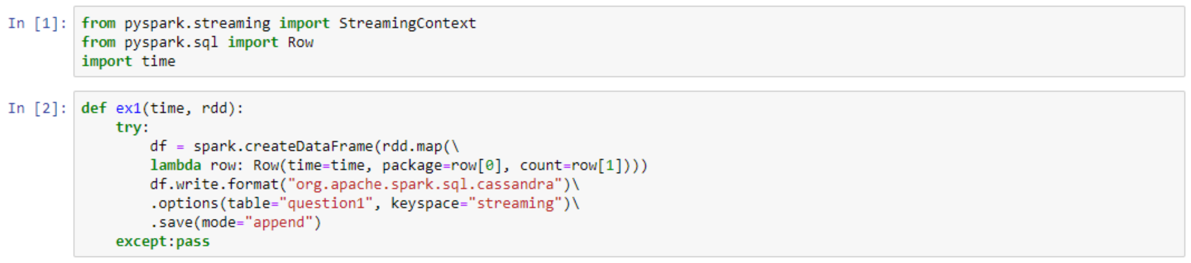
Big Data Management
Assignment 3


Home Page - Select or create a n
assignment3 - Jupyter Notebook
Not secure | 192.168.56.200:8888/notebooks/assignment3.ipynb
jupyter assignment3 Last Checkpoint: a day ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

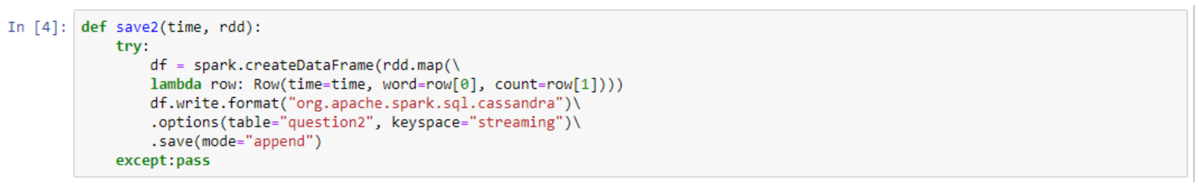
```
In [1]: vehicle_counter_DF = spark.read.csv('/home/amani/per-vehicle-records-2019-10-31.csv',inferSchema = True, header = True)

In [2]: type(vehicle_counter_DF)
Out[2]: pyspark.sql.dataframe.DataFrame

In [3]: vehicle_counter_DF.count()
Out[3]: 4389299
```



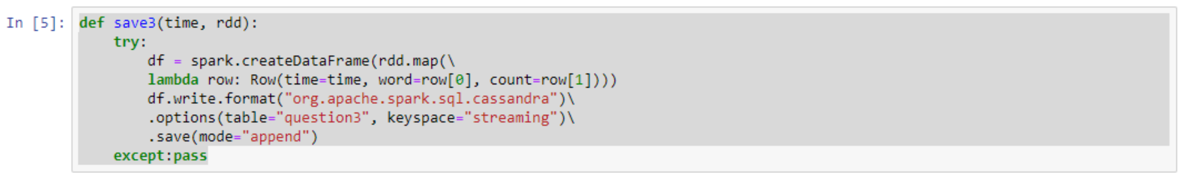
```
In [1]: vehicle_counter_DF = spark.read.csv('/home/amani/per-vehicle-records-2019-10-31.csv',inferSchema = True, header = True)

In [2]: type(vehicle_counter_DF)
Out[2]: pyspark.sql.dataframe.DataFrame

In [3]: vehicle_counter_DF.count()
Out[3]: 4389299
```



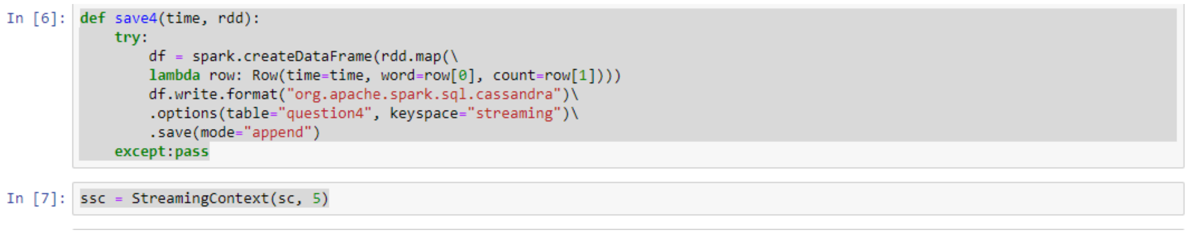
```
In [1]: from pyspark.streaming import StreamingContext
        from pyspark.sql import Row
        import time

In [2]: def ex1(time, rdd):
            try:
                df = spark.createDataFrame(rdd.map(\
                lambda row: Row(time=time, package=row[0], count=row[1])))
                df.write.format("org.apache.spark.sql.cassandra")\
                .options(table="question1", keyspace="streaming")\
                .save(mode="append")
            except:pass
```



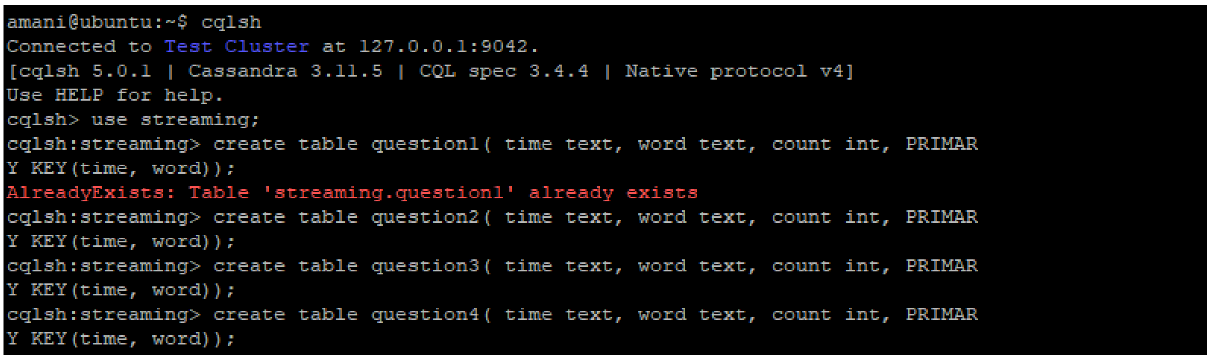
```
In [4]: def save2(time, rdd):
            try:
                df = spark.createDataFrame(rdd.map(\
                lambda row: Row(time=time, word=row[0], count=row[1])))
                df.write.format("org.apache.spark.sql.cassandra")\
                .options(table="question2", keyspace="streaming")\
                .save(mode="append")
            except:pass
```



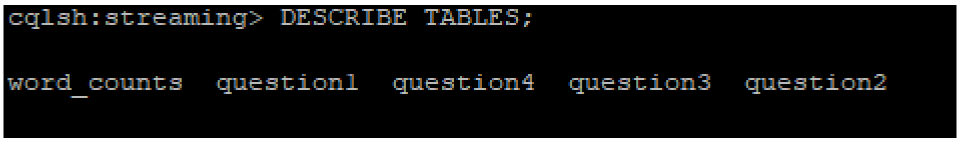
```
In [5]: def save3(time, rdd):
            try:
                df = spark.createDataFrame(rdd.map(\
                lambda row: Row(time=time, word=row[0], count=row[1])))
                df.write.format("org.apache.spark.sql.cassandra")\
                .options(table="question3", keyspace="streaming")\
                .save(mode="append")
            except:pass
```



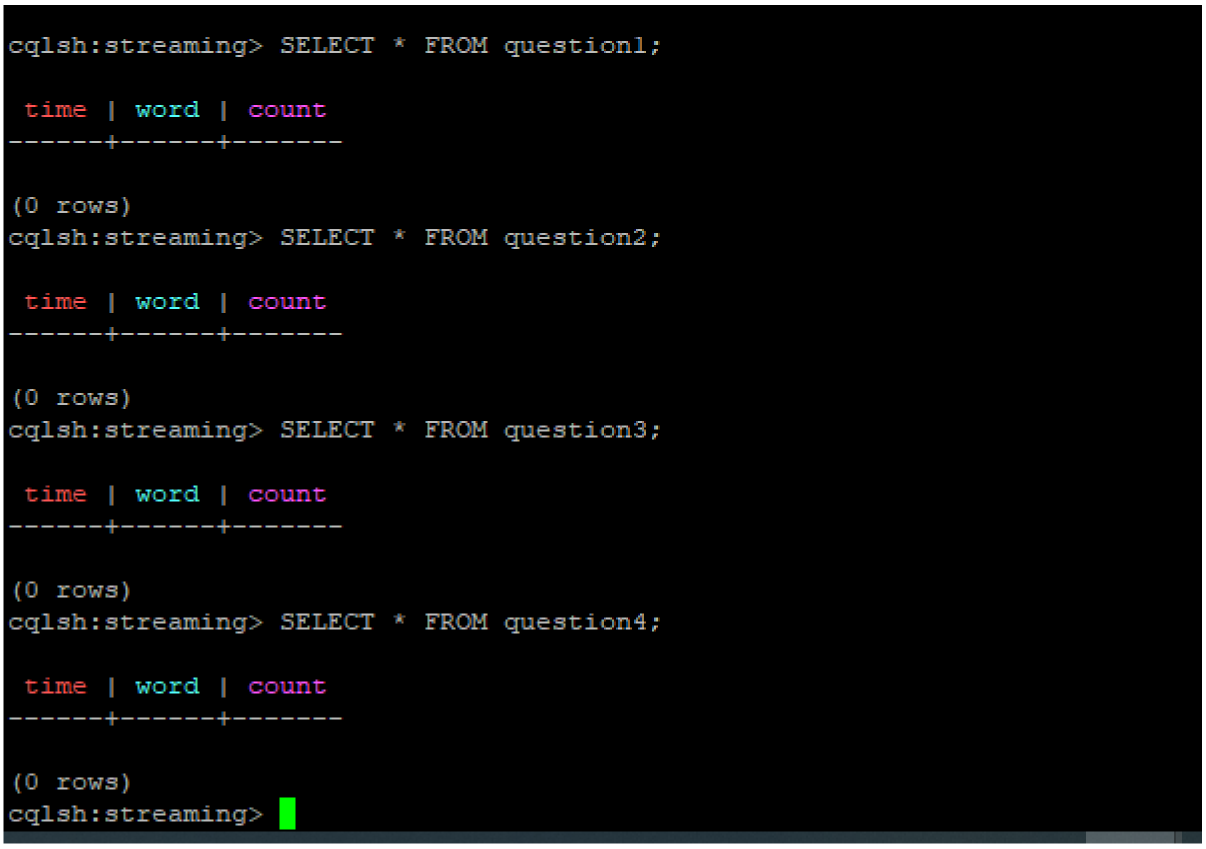
```
In [6]: def save4(time, rdd):
            try:
                df = spark.createDataFrame(rdd.map(\
                lambda row: Row(time=time, word=row[0], count=row[1])))
                df.write.format("org.apache.spark.sql.cassandra")\
                .options(table="question4", keyspace="streaming")\
                .save(mode="append")
            except:pass

In [7]: ssc = StreamingContext(sc, 5)
```



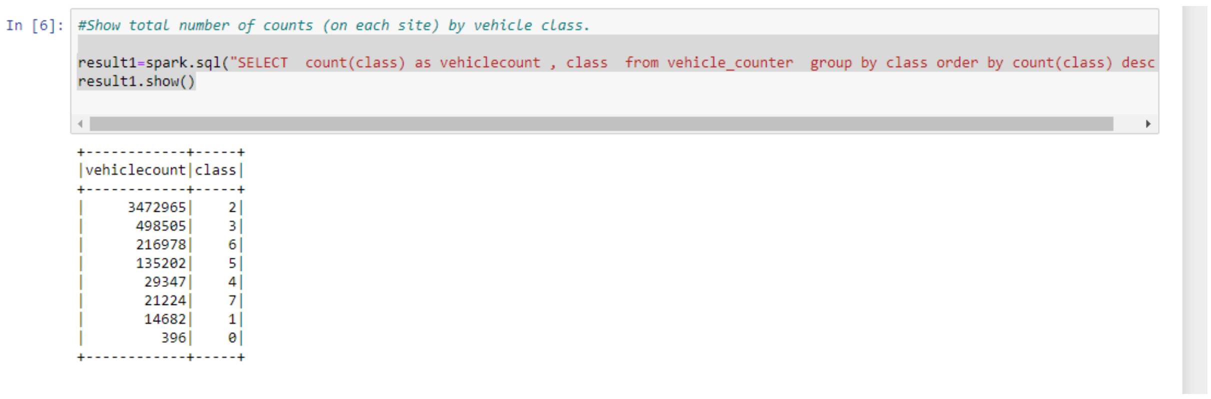
```
amani@ubuntu:~$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.5 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> use streaming;
cqlsh:streaming> create table question1( time text, word text, count int, PRIMAR
Y KEY(time, word));
AlreadyExists: Table 'streaming.question1' already exists
cqlsh:streaming> create table question2( time text, word text, count int, PRIMAR
Y KEY(time, word));
cqlsh:streaming> create table question3( time text, word text, count int, PRIMAR
Y KEY(time, word));
cqlsh:streaming> create table question4( time text, word text, count int, PRIMAR
Y KEY(time, word));
```



```
cqlsh:streaming> DESCRIBE TABLES;

word_counts  question1  question4  question3  question2
```

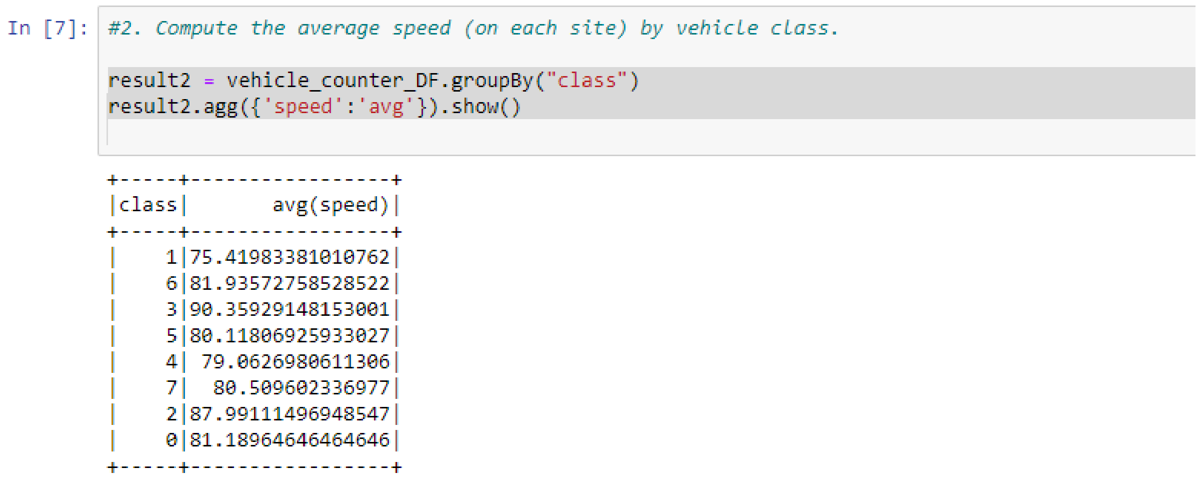


```
cqlsh:streaming> SELECT * FROM question1;

 time | word | count
------+------+-------

(0 rows)
cqlsh:streaming> SELECT * FROM question2;

 time | word | count
------+------+-------

(0 rows)
cqlsh:streaming> SELECT * FROM question3;

 time | word | count
------+------+-------

(0 rows)
cqlsh:streaming> SELECT * FROM question4;

 time | word | count
------+------+-------

(0 rows)
cqlsh:streaming>
```



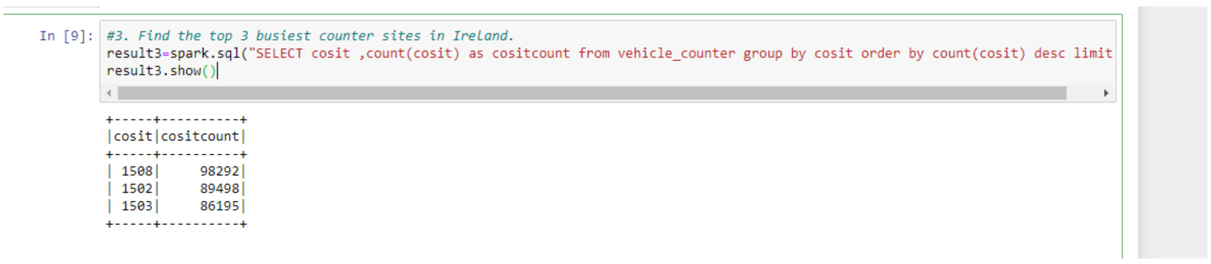
```
In [6]: #Show total number of counts (on each site) by vehicle class.

        result1=spark.sql("SELECT  count(class) as vehiclecount , class  from vehicle_counter  group by class order by count(class) desc
        result1.show()
```

```
+------------+-----+
|vehiclecount|class|
+------------+-----+
|     3472965|    2|
|      498505|    3|
|      216978|    6|
|      135202|    5|
|       29347|    4|
|       21224|    7|
|       14682|    1|
|         396|    0|
+------------+-----+
```



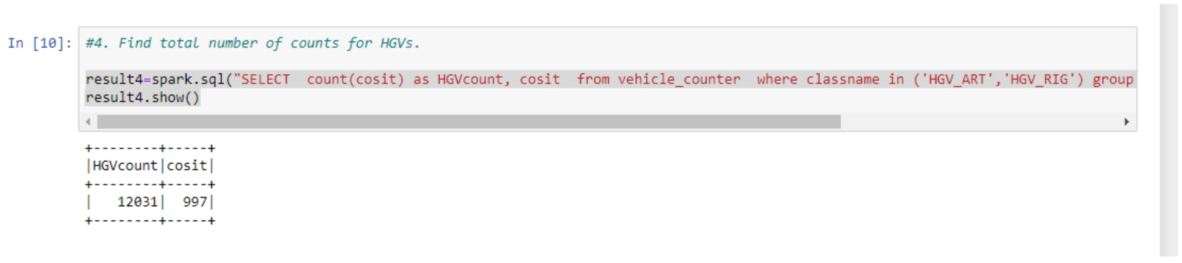
```
In [7]: #2. Compute the average speed (on each site) by vehicle class.

        result2 = vehicle_counter_DF.groupBy("class")
        result2.agg({'speed':'avg'}).show()
```

```
+-----+-----------------+
|class|       avg(speed)|
+-----+-----------------+
|    1|75.41983381010762|
|    6|81.93572758528522|
|    3|90.35929148153001|
|    5|80.11806925933027|
|    4| 79.0626980611306|
|    7|  80.509602336977|
|    2|87.99111496948547|
|    0|81.18964646464646|
+-----+-----------------+
```



```
In [9]: #3. Find the top 3 busiest counter sites in Ireland.
        result3=spark.sql("SELECT cosit ,count(cosit) as cositcount from vehicle_counter group by cosit order by count(cosit) desc limit
        result3.show()
```

```
+-----+----------+
|cosit|cositcount|
+-----+----------+
| 1508|     98292|
| 1502|     89498|
| 1503|     86195|
+-----+----------+
```



```
In [10]: #4. Find total number of counts for HGVs.

         result4=spark.sql("SELECT  count(cosit) as HGVcount, cosit  from vehicle_counter  where classname in ('HGV_ART','HGV_RIG') group
         result4.show()
```

```
+--------+-----+
|HGVcount|cosit|
+--------+-----+
|   12031|  997|
+--------+-----+
```