

Assignment 2

Big Data - Spring 2018

Using Spark to explore NYC Parking Violations

In this assignment, we will analyze different aspects of parking violations in NYC using Spark and python. You will write pyspark programs using both RDDs and DataFrames for 7 different tasks, described later in this document.

We will use the same datasets used in Assignment 1. As in Assignment 1, the full parking-violations.csv and open-violations.csv files for the March 2016 dataset have already been made accessible on HDFS on dumbo for you - **you should not store additional copies in your HDFS directory**. They are located at

```
/user/ecc290/HW1data/parking-violations.csv
```

and

```
/user/ecc290/HW1data/open-violations.csv
```

We have also made accessible versions with the headers included, which will be useful for defining schema in the SparkSQL tasks:

```
/user/ecc290/HW1data/parking-violations-header.csv
```

```
/user/ecc290/HW1data/open-violations-header.csv
```

And again, for debugging purposes, we advise you to first test and debug your code on a smaller dataset, which you should create yourself from the available data.

For your final submission, you will run the Spark programs on the complete March 2016 datasets using Spark on dumbo.

Submission:

You will submit the Spark programs for all tasks in a .zip file named "yournetid.zip" (e.g., "ecc290.zip"). The zip file should include 14 python files: task1.py, task1-sql.py,...,task7.py, task7-sql.py.

***For this assignment, you do not need to include output files in your submission.**

You should also not include any input files in your submission.



Notes:

- For each task you will use both core Spark using RDDs and SparkSQL using DataFrames.
- Your program should read in the path to the input file on HDFS from the command line arguments. **For task 1, you are guaranteed that parking-violations.csv will be the first of the two files passed in.** In other words, we will execute your programs with the following commands:

For task 1:

```
spark-submit --conf
spark.pyspark.python=/share/apps/python/3.4.4/bin/python
{your_netid}/task1.py /user/ecc290/HW1data/parking-violations.csv
/user/ecc290/HW1data/open-violations.csv
```

For task 1 - SQL:

```
spark-submit --conf
spark.pyspark.python=/share/apps/python/3.4.4/bin/python
{your_netid}/task1-sql.py /user/ecc290/HW1data/parking-violations-
header.csv /user/ecc290/HW1data/open-violations-header.csv
```

For task 3:

```
spark-submit --conf
spark.pyspark.python=/share/apps/python/3.4.4/bin/python
{your_netid}/task3.py /user/ecc290/HW1data/open-violations.csv
```

For task 3-SQL:

```
spark-submit --conf
spark.pyspark.python=/share/apps/python/3.4.4/bin/python
{your_netid}/task3-sql.py /user/ecc290/HW1data/open-violations-
header.csv
```

For all other tasks:

```
spark-submit --conf
spark.pyspark.python=/share/apps/python/3.4.4/bin/python
{your_netid}/taskx.py /user/ecc290/HW1data/parking-violations.csv
```

and for SQL versions:

```
spark-submit --conf
spark.pyspark.python=/share/apps/python/3.4.4/bin/python
{your_netid}/taskx-sql.py /user/ecc290/HW1data/parking-violations-
header.csv
```

- You should only use the most recent available versions of python and Spark on dumbo (3.4.4 and 2.2.0, respectively)
- Your code should output a directory named “taskx.out” or “taskx-sql.out” to HDFS, i.e., **use the Spark RDD function `saveAsTextFile(“taskx.out”)` or the Spark DataFrame function `save(“taskx-sql.out”,format=“text”)`** rather than python I/O. (In order for the hw2tester script to work, you must name your output directories “taskx.out” and “taskx-sql.out” where x is in 1 through 7).

- As in Assignment 1, you will be reading from CSV files. To do this in core Spark, reading into an RDD, you would use, e.g.,

```
from csv import reader

lines = sc.textFile(sys.argv[1], 1)

lines = lines.mapPartitions(lambda x: reader(x))
```

To do this in SparkSQL, reading into a DataFrame, you would use, e.g.,

```
parking =
spark.read.format('csv').options(header='true',inferSchema='true').load
(sys.argv[1])
```

- For core Spark: You may find that using a final `map()` stage is helpful for formatting your output correctly.
- For SparkSQL: You may find that using a final `select()` which produces a single column using `format_string()` is helpful for formatting the output and writing to a text file. For example, for Task 1-SQL, if *result* is the dataframe that contains the query results, you could write:


```
result.select(format_string('%d\t%s, %d, %d, %s',result.summons_number,result.plate_id,result.violation_precinct,result.violation_code,date_format(result.issue_date,'yyyy-MM-dd'))).write.save("task1-sql.out",format="text")
```

Assignment:

=====

Task 1

Write a Spark program that finds all parking violations that have been paid, i.e., that do not occur in open-violations.csv.

Output: A key-value pair per line, where

key = summons_number

values = plate_id, violation_precinct, violation_code, issue_date

You should separate the key and value by a tab character ('\t') and elements within the key/value should be separated by a comma and a space. Your output format should conform to the following example:

```
4617117696      GRV2608, 0, 36, 2016-03-09
4617863450      HAM2650, 0, 36, 2016-03-24
```

Task 1-SQL

Write a SparkSQL program for Task 1.

=====

Task 2

Write a Spark program that finds the distribution of the violation types, i.e., for each violation code, the number of violations that have this code.

Output: A key-value pair per line, where

key = violation_code

value = number of violations

You should separate the key and value by a tab character ('\t') and elements within the key/value should be separated by a comma and a space. Your output format should conform to the following example:

```
1      159
2      5
```

Task 2-SQL

Write a SparkSQL program for Task 2.

=====

Task 3

Write a Spark program that finds the total and average amount due in open violations for each license type.

Output: A key-value pair per line, where

key = license_type

value = total, average

where total and average are rounded to 2 decimal places.

You should separate the key and value by a tab character ('\t') and elements within the key/value should be separated by a comma and a space. Your output format should conform to the following example:

```
PAS      9482469.38, 35.82
```

```
USC      250.00, 125.00
```

Task 3-SQL

Write a SparkSQL program for Task 3.

=====

Task 4

Write a Spark program that computes the total number of violations for vehicles registered in the state of NY and all other vehicles.

Output: 2 key-value pairs with one key-value pair per line.

You should separate the key and value by a tab character ('\t'). Your output format should conform to the following example:

```
NY      12345
```

```
Other   6789
```

Task 4-SQL

Write a SparkSQL program for Task 4.

=====

Task 5

Write a Spark program that finds the vehicle that has had the greatest number of violations (assume that plate_id and registration_state uniquely identify a vehicle).

Output: One key-value pair

You should separate the key and value by a tab character ('\t') and elements within the key/value should be separated by a comma and a space. Your output format should conform to the following example:

```
AP501F, NJ      138
```

Task 5-SQL

Write a SparkSQL program for Task 5.

=====

Task 6

Write a Spark program that finds the top-20 vehicles in terms of total violations (assume that plate id and registration state uniquely identify a vehicle).

Output: List of 20 key-value pairs, ordered by decreasing number of violations. For items with the same number of violations, order by ascending plate_id.

You should separate the key and value by a tab character ('\t') and elements within the key/value should be separated by a comma and a space. Your output format should conform to the following example:

```
AP501F, NJ      138
```

Task 6-SQL

Write a SparkSQL program for Task 1.

=====

Task 7

In March 2016, the 5th, 6th, 12th, 13th, 19th, 20th, 26th, and 27th were weekend days (i.e., Sat. and Sun.).

Write a Spark program that, for each violation code, lists the average number of violations with that code issued per day on weekdays and weekend days. You may hardcode "8" as the number of weekend days and "23" as the number of weekdays in March 2016.

Output: List of key-value pairs where

```
key = violation_code
```

```
value = weekend_average, week_average
```

where weekend_average and week_average are rounded to 2 decimal places.

You should separate the key and value by a tab character ('\t') and elements within the key/value should be separated by a comma and a space. Your output format should conform to the following example:

```
1      3.25, 5.78
```

```
2      0.12, 0.17
```

Task 7-SQL

Write a SparkSQL program for Task 7.

=====

Testing your solutions

We have provided a script to test your solutions on the March 2016 data. On dumbo, type

```
hfs -get /user/ecc290/HW1data/hw2tester.tar
```

```
hfs -get /user/ecc290/HW1data/README
```

To unpack the files, type

```
tar xvf hw2tester.tar
```

To run the tests, type

```
./testall.sh <INPUTPATH>
```

where <INPUTPATH> is the **full** path to your directory containing task1.py, task1-sql.py, task2.py, task2-sql.py, etc. **without a trailing slash** (e.g., "/home/ecc290")

You will be told for each task, whether you pass or fail. If you fail some task X, you can view the diff of your output and the solution file in results/taskX.diff.

PLEASE READ THE README FILE FOR FURTHER DETAILS!