HW1 solutions

You can find this ppt in resources/labs/HW1 sol/ HW1 solutions.pdf

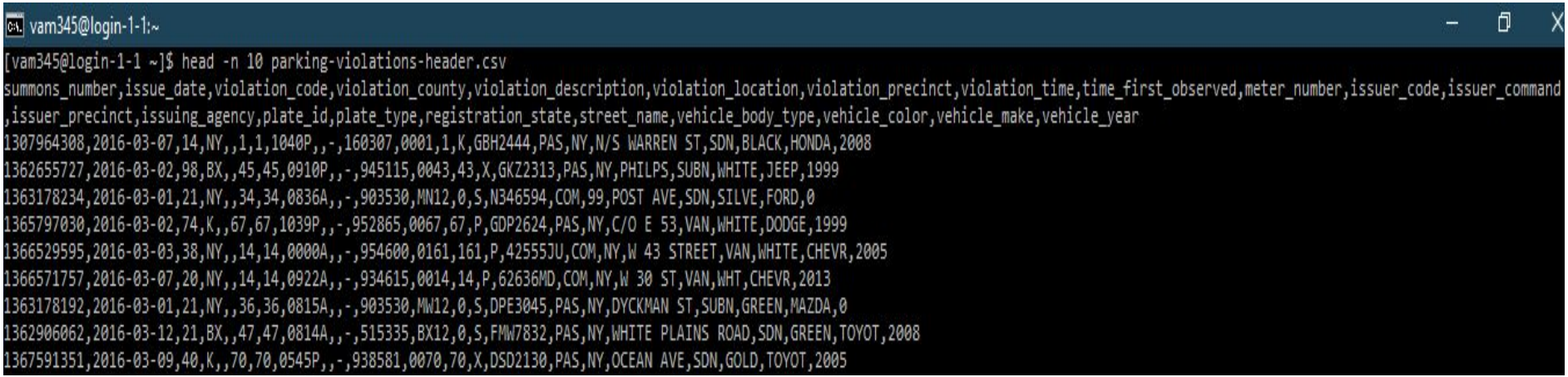
Task1

Find all parking violations that have been paid, i.e., that do not occur in openviolations.csv.

Output: A key-value* pair per line, where: key = summons_number, values = plate_id, violation_precinct, violation_code, issue_date

Task1 ---> Map Reduce

- Get both the files and see the contents of those files (First 10 lines of the files)
- head -n 10 open-violations-header.csv
- head -n 10 parking-violations-header.csv



Task1 ---> Map Reduce (map.py)

```
for line in sys.stdin:
    # Get the filename which the mapper is processing
    fn = os.environ.get("mapreduce_map_input_file")
    entry = next(reader([line]))
    summonsnum = entry[0]
    if "open" in fn:
        opv = 1
        print('{0:s}\t{1:d}'.format(summonsnum,opv))
```

Task1 ---> Map Reduce (map.py) Contd.

```
else:
        opv = 0
        plate_id = entry[14]
        violation_precinct = entry[6]
        violation_code = entry[2]
        issue_date = entry[1]
        print('{0:s}\t{1:d},{2:s}, {3:s}, {4:s},
{5:s}'.format(summonsnum,opv,plate_id, violation_precinct,
violation_code, issue_date))
```

Task1 ---> Map Reduce (reduce.py)

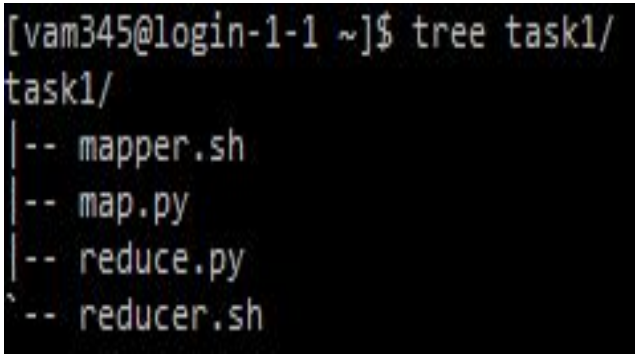
```
currentkey = None
paid =1
curval = None

# input comes from STDIN (stream data that goes to the
program)
for line in sys.stdin:

    #Remove leading and trailing whitespace
    line = line.strip()

    #Get key/value
    key, value = line.split('\t',1)
    vl = value.split(',',1)
    opv = vl[0]
    if int(opv)==0:
        val = vl[1]
```

Task1 ---> Map Reduce (reduce.py) contd.

```
    if key==currentkey:
        if int(opv)==1:
            paid =0
        else:
            curval = val
     else:
        if currentkey:
            if (paid == 1):
                print('{0:s}\t{1:s}\'.format(currentkey,curval))
        currentkey = key
        paid = 1
        if int(opv)==1:
            paid = 0
        else:
            curval = val
if currentkey:
    if (paid == 1):
        print('{0:s}\t{1:s}'.format(currentkey,curval))
```

Task1 ---> Map Reduce (Running the program)

Directory Structure



mapper.sh

```
#!/bin/bash

. /etc/profile.d/modules.sh
module load python/gnu/3.4.4
task1/map.py
```

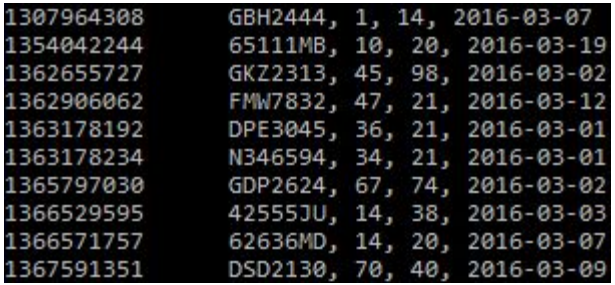
reducer.sh

```
#!/bin/bash

. /etc/profile.d/modules.sh
module load python/gnu/3.4.4
task1/reduce.py
```

```
hjs -D mapreduce.job.reduces=2 -file ~/task1 -mapper task1/mapper.sh -reducer
task1/reducer.sh -input /user/ecc290/HW1data/parking-violations.csv -input
/user/ecc290/HW1data/open-violations.csv -output /user/vam345/task1/task1.out
```

Task1 ---> Map Reduce (Debug)

```
head -n 10 -q open-violations.csv parking-violations.csv | python task1/map.py | sort -k1,1 | python task1/reduce.py
```



Important Point :

- Make sure, open-violations and parking-violations.csv files are in local (Not in HDFS).
- This above command is only for checking whether your syntax is correct or not.

Task1 (spark)

```
# custom function
def redfunc(p,k):
    sum = p[0]+ k[0]
    if sum == 0:
        return p
    else:
        return (1,1)

parking = sc.textFile(sys.argv[1], 1)
parking = parking.mapPartitions(lambda x: reader(x))

open = sc.textFile(sys.argv[2],1)
open = open.mapPartitions(lambda x: reader(x))

opensums = open.map(lambda x: (x[0], (1,1)))
parkingvs = parking.map(lambda x: (x[0], (0,x[14],x[6],x[2],x[1])))
all = sc.union([opensums,parkingvs])
ps = all.reduceByKey(redfunc)
closed = ps.filter(lambda line: len(line[1])==5)
```

Task1 (spark-sql)

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

diff = spark.sql(
"SELECT summons_number FROM parking EXCEPT (SELECT summons_number FROM open)"
)
diff.createOrReplaceTempView("diff")

filtpark = spark.sql(
"SELECT parking.summons_number, plate_id, violation_precinct, violation_code, \
issue_date FROM parking, diff WHERE parking.summons_number = diff.summons_number \
ORDER BY parking.summons_number"
)
```

Task 2

Find the frequencies of the violation types in parking_violations.csv, i.e., for each violation code, the number of violations that this code has.

Output: A key-value pair per line, where:

key = violation_code

value = number of violations

Task2 (map reduce) map.py

```
from csv import reader
for line in sys.stdin:
    entry = next(reader([line]))
    violation_code = entry[2]
    print('{0:s}\t{1:d}'.format(violation_code,1))
```

Task2 (map reduce) reduce.py

```
currentkey = None
cursum = 0
for line in sys.stdin:
    line = line.strip()
    key, value = line.split('\t',1)

    #If we are still on the same key...
    if key==currentkey:
        cursum += int(value)
    #Otherwise, if this is a new key...
    else:
        #If this is a new key and not the first key we've seen
        if currentkey:
            print('{0:s}\t{1:d}'.format(currentkey, int(cursum)))

        currentkey = key
        cursum = int(value)

if currentkey:
    print('{0:s}\t{1:d}'.format(currentkey,int(cursum)))
```

Task2 (spark)

```
from operator import add
from csv import reader
parking = sc.textFile(sys.argv[1], 1)
parking = parking.mapPartitions(lambda x: reader(x))

violations = parking.map(lambda x: (int(x[2]), 1))
violationnums = violations.reduceByKey(add)

# formating string
vformat = violationnums.map(lambda x:
'{0:s}\t{1:d}'.format(str(x[0]),int(x[1])) )
vformat.saveAsTextFile("task2.out")
```

Task2 (spark-sql)

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

dff = spark.sql(
     "select violation_code as vv, count(*) as cc from parking group by violation_code"
)
dff = dff.sort("vv")

dff.select(
     format_string('%d\t%d',dff.vv,dff.cc)
).write.save("task2-sql.out",format="text")
```

Task 3

Find the total and average amounts 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.

Task3 (map reduce) map.py

```
for line in sys.stdin:

    entry = next(reader([line]))
    license_type = entry[2];
    if entry[12].strip()=='':
        amt_due = 0.0
    else:
        amt_due = float(entry[12])
    print('{0:s}\t{1:f}'.format(license_type,amt_due))
```

Task3 (map reduce) reduce.py

```
currentkey = None
curtotal = 0.0
curcount = 0

for line in sys.stdin:
    line = line.strip()
    key, value = line.split('\t',1)

    #If we are still on the same key...
    if key==currentkey:
        curcount = curcount +1
        curtotal += float(value)
    #Otherwise, if this is a new key...
    else:
        #If this is a new key and not the first key we've seen
        if currentkey:
            print('{0:s}\t{1:.2f}, {2:.2f}'.format(
                      currentkey, float(curtotal), float(curtotal)/curcount))
        currentkey = key
        curtotal = float(value)
        curcount = 1
if currentkey:
    print('{0:s}\t{1:.2f}, {2:.2f}'.format(currentkey, float(curtotal), float(curtotal)/curcount))
```

Task3 (spark)

```
def totavg(p,k):
    tot = float(p[0])+float(k[0])
    cnt = int(p[1])+int(k[1])
    return (tot, cnt)

def formt(x):
    return '{0:s}\t{1:.2f}, {2:.2f}'.format(
              str(x[0]),float(x[1][0]), float(x[1][0])/float(x[1][1]))

if __name__ == "__main__":
    sc = SparkContext()
    open = sc.textFile(sys.argv[1], 1)
    open = open.mapPartitions(lambda x: reader(x))
    entry = open.map(lambda x: (x[2], (x[12],1)))
    tc = entry.reduceByKey(totavg)
    ta = tc.map(lambda x: formt(x))
    ta.saveAsTextFile("task3.out")
```

Task3 (spark-sql)

```
open = spark.read.format('csv').options(
    header='true',inferschema='true').load(sys.argv[1])
open.createOrReplaceTempView("open")

amts = spark.sql("select license_type as ll, sum(amount_due) as ss,\
    sum(amount_due)/count(*) as aa from open group by license_type")
amts = amts.sort("ll")

dff = amts.select(format_string('%s\t%.2f, %.2f', amts["ll"], amts["ss"],
amts["aa"])).write.save("task3-sql.out",format="text")
```

Task 4

Compute 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, following the key-value format below:

NY <total number>

Other <total number>

Task4 (spark)

```
def nyo(p):
    if 'ny' in p.lower():
        return 'NY'
    else:
        return 'Other'


if __name__ == "__main__":
    sc = SparkContext()
    parking = sc.textFile(sys.argv[1], 1)
    parking = parking.mapPartitions(lambda x: reader(x))

    entry = parking.map(lambda x: (nyo(x[16]),1))
    tc = entry.reduceByKey(add)
    tc = tc.map(lambda x: '{0:s}\t{1:d}'.format(str(x[0]), int(x[1])))
    tc.saveAsTextFile("task4.out")
```

Task4 (spark-sql)

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

parking.createOrReplaceTempView("parking")
parking_ny = spark.sql("SELECT registration_state FROM parking \
    WHERE registration_state='NY'")

parking_other = spark.sql("SELECT registration_state FROM parking \
    WHERE registration_state<>'NY'")

parking_ny.createOrReplaceTempView("parking_ny")
parking_other.createOrReplaceTempView("parking_other")

ny_count = spark.sql("SELECT 'NY' AS state, count(*) as count FROM parking_ny")
other_count = spark.sql("SELECT 'Other' AS state, count(*) as count FROM parking_other")

union = ny_count.union(other_count)
union.select(format_string('%s\t%d',union["state"],union["count"])).write.save(
    "task4-sql.out",format="text")
```

Task 5

Find 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, following the key-value format below:

plate_id, registration_state <total_number>

Task5 (spark)

```
parking = sc.textFile(sys.argv[1], 1)
parking = parking.mapPartitions(lambda x: reader(x))

entry = parking.map(lambda x: ( (x[14],x[16])  ,1))
tc = entry.reduceByKey(add)
top = sc.parallelize(tc.sortBy(lambda x: x[1], False).take(1))
top = top.map(
    lambda x:'{0:s}, {1:s}\t{2:d}'.format(str(x[0][0]),str(x[0][1]),int(x[1]))
    )
top.saveAsTextFile("task5.out")
```

Task5 (spark-sql)

```
p2 = spark.sql("select parking.plate_id as pid,\
    parking.registration_state as rs,count(*) as cc from parking \
    group by parking.plate_id,parking.registration_state")

p2.createOrReplaceTempView("p2")

pp2 = spark.sql("select p.pid as pid, p.rs as rs, p.cc as mc \
    from p2 p where p.cc = (select max(pp.cc) from p2 pp)")

pp2.select(format_string('%s,%s\t%d',pp2["pid"],pp2["rs"],pp2["mc"])).write.save(
    "task5-sql.out",format="text")
```

Task 6

Find 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 using the following format: plate_id, registration_state

*Ordered by decreasing number of violations. For items with the same number of violations, order by ascending plate_id.

Task6 (spark)

```
parking = sc.textFile(sys.argv[1], 1)
parking = parking.mapPartitions(lambda x: reader(x))

entry = parking.map(lambda x: ( (x[14],x[16]), 1))
tc = entry.reduceByKey(add)

top = sc.parallelize(
    tc.sortBy(lambda x: x[0][0],True).sortBy(lambda x: x[1], False).take(20)
    )

top = top.map(
    lambda x: '{0:s}, {1:s}\t{2:d}'.format(str(x[0][0]),str(x[0][1]),int(x[1]))
    )
top.saveAsTextFile("task6.out")
```

Task6 (spark-sql)

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

parking.createOrReplaceTempView("parking")

p2 = spark.sql("select parking.plate_id as pid,\
    parking.registration_state as rs,count(*) as cc from \
    parking group by parking.plate_id,parking.registration_state \
    order by cc DESC, parking.plate_id asc limit 20")
```

Task 7

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

Output: List of key-value pairs using the following format:

violation_code weekend_average, week_average

where weekend_average and week_average are rounded to 2 decimal places

Task7 (spark)

```
def iswknd(x,wknds):
    y,m,d = x.split('-')
    day = int(d)
    if day in wknds:
        return 1
    else:
        return 0

def redfunc(p,k):
    return (p[0]+k[0], p[1]+k[1])
```

Task7 (spark) contd.

```
if __name__ == "__main__":
    sc = SparkContext()
    dts = [5,6,12,13,19,20,26,27]

    parking = sc.textFile(sys.argv[1], 1)
    parking = parking.mapPartitions(lambda x: reader(x))
    entry = parking.map(
         lambda x: ( x[2], (iswknd(x[1],dts), not iswknd(x[1],dts) ) )
         )
    tc = entry.reduceByKey(redfunc)
    tc = tc.map(lambda x: '{0:d}\t{1:.2f}, {2:.2f}'.format(int(x[0]),
         float(x[1][0])/8.0, float(x[1][1])/23.0 ))
    tc.saveAsTextFile("task7.out")
```

Task7 (spark-sql)

```
spark.udf.register("isWeekend",
     lambda x: float(1.0/8.0) if x in [5,6,12,13,19,20,26,27] else float(0.0))
spark.udf.register("isntWeekend",
     lambda x: float(1.0/23.0) if x not in [5,6,12,13,19,20,26,27] else float(0.0))

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

p2 = spark.sql("select parking.violation_code as vc, DAY(parking.issue_date) as day from
parking")

p2.createOrReplaceTempView("p2")

p4 = spark.sql("select vc as vc, sum(isWeekend(day)) as wd, sum(isntWeekend(day)) as wk
from p2 group by vc order by vc")

p4.select(format_string('%d\t%.2f,%.2f',p4.vc,p4.swd,p4.swk)
          ).write.save("task7-sql.out",format="text")
```