University of Augsburg, Institute of Computer Science
Prof. Dr. P. Fischer
J. Kastner, L. Rudenko

SS 2019
10. June 2019
Solution 5

Analyzing Massive Data Sets

Exercise 1: Spark (homework)

```
from operator import add
from pysparkling import Context

sc = Context()
```

a) Find 25 suppliers with the lowest account balance.

```
top_25 = (
    sc.textFile('supplier.tbl')
    .map(lambda line: line.split('|'))
    .map(lambda row: (row[1], float(row[5])))
    .sortBy(lambda row: row[1])
    .take(25)
)

# more efficient on PySpark (without global sorting, which causes substantial
# shuffle/repartitioning), exactly the same as above solution on Pysparkling,
# which implements top(num, key) as sortBy(key, ascending=False).take(num)
top_25 = (
    sc.textFile('supplier.tbl')
    .map(lambda line: line.split('|'))
    .map(lambda row: (row[1], float(row[5])))
    .top(25, key=lambda row: -row[1])
    # or .takeOrdered(25, key=lambda row: row[1]) # only available on PySpark
)

print(top_25)
```

b) How many suppliers have a positive account balance?

```
num_pos_bal = (
    sc.textFile('supplier.tbl')
    .map(lambda line: float(line.split('|')[5]))
    .filter(lambda acctbal: acctbal > 0)
    .count()
)

print(num_pos_bal)
```

c) Find out all brands produced by the same manufacturer and calculate the items number and the total sales price for each brand of each manufacturer.

```
brand_mfgr_count = (
    sc.textFile('part.tbl')
    .map(lambda line: line.split('|'))
    .map(lambda row: ((row[2], row[3]), 1))
    .reduceByKey(add)
)

brand_mfgr_sum = (
    sc.textFile('part.tbl')
    .map(lambda line: line.split('|'))
    .map(lambda row: ((row[2], row[3]), float(row[7])))
    .reduceByKey(add)
)
```

```
result = brand_mfgr_count.join(brand_mfgr_sum).sortBy(lambda x: x[0]).collect()
print(result)

# shorter solution
def add_tuples_elementwise(*args):
    return tuple(sum(x) for x in zip(*args))

result = (
    sc.textFile('part.tbl')
    .map(lambda line: line.split('|'))
    .map(lambda row: ((row[2], row[3]), (1, float(row[7]))))
    .reduceByKey(add_tuples_elementwise)
)

print(result.sortBy(lambda x: x[0]).collect())
```

d) How many items have 3 words in their name?

```
num_name_length_3 = (
    sc.textFile('part.tbl')
    .map(lambda line: line.split('|'))
    .map(lambda row: len(row[1].split()))
    .filter(lambda length: length == 3)
    .count()
)

print(num_name_length_3)
```

e) How many different items does each supplier have?

```
supplier = (
    sc.textFile('supplier.tbl')
    .map(lambda line: line.split('|'))
    .map(lambda row: (row[0], (row[1], row[2], row[3], row[4], row[5], row[6])))
)

part_supplier_s = (
     sc.textFile('partsupp.tbl')
     .map(lambda line: line.split('|'))
     .map(lambda row: (row[1], (row[0], row[2], row[3], row[4])))

)

supplier_ps_right = (
    supplier.rightOuterJoin(part_supplier_s)
    .map(lambda x: (x[1][1][0], (x[1][0], x[0], x[1][1][1:])))
)

part = (
   sc.textFile('part.tbl')
   .map(lambda line: line.split('|'))
   .map(lambda row: (row[0], (row[1], row[2], row[3], row[4], row[5], row[6], row[7], row[8])))

)

supplier_part = (
    supplier_ps_right.rightOuterJoin(part)
    .map(lambda x: (x[1][0][0][0], 1))
    .reduceByKey(add)
)

print(supplier_part.collect())
```

Note: The is *.join* on Pysparkling, but it works incorrectly. The correct result for our query e) can be obtained using *.rightOuterJoin*, but in the general case that won't work.

It is possible to express and answer all 5 queries with Spark's RDDs. For the queries c) and e), however, the result is very elaborate and unattractive. **DataFrame** concept from Spark offers a better approach.

Exercise 2: Spark DataFrames (live)

The solution was discussed in the exercise.

Exercise 3: MinHashing (live)

The solution was discussed in the exercise.