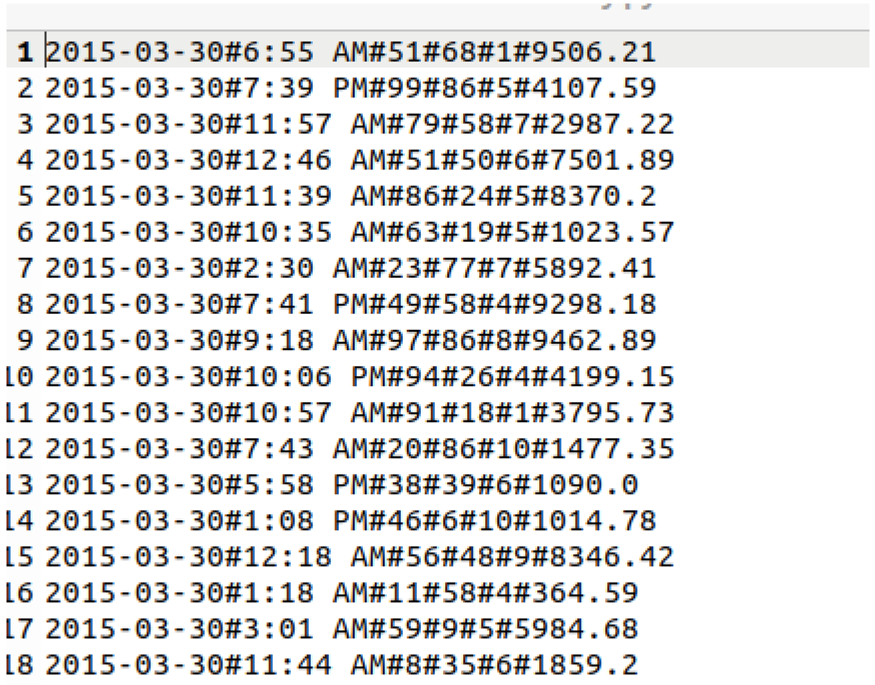
Zadatak 2, Spark Core API

- Napisati program koji na osnovnu obavljenih transakcija i određenog skupa pravila klijentima dodjeljuje nagradne proizvode
  - Klijentu koji je napravio najviše transakcija kao poklon dodjeljuje se proizvod ID=4
  - Klijentima koji su kupili >=2 proizvoda ID=25 obračunati 5% popusta
  - Klijentima koji su kupili >=5 proizvoda ID=81 pokloniti proizvod ID=70
  - Klijentu koji je potrošio najviše novca poklon je proizvod ID=63
  - Nagradni proizvodi upisuju se kao nove transakcije čija je vrijednost 0
  - Prikazati listu transakcija nagrađenih klijenata

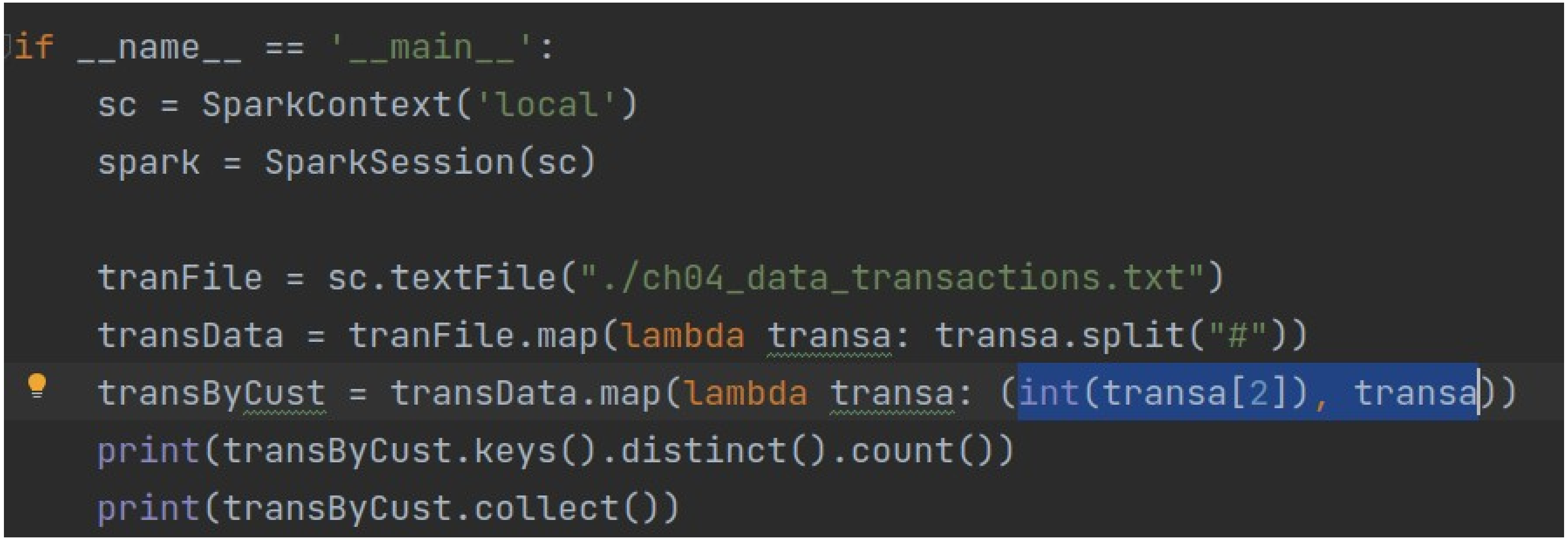
Dataset

- Transaction date, time, customer ID, product ID, quantity, price



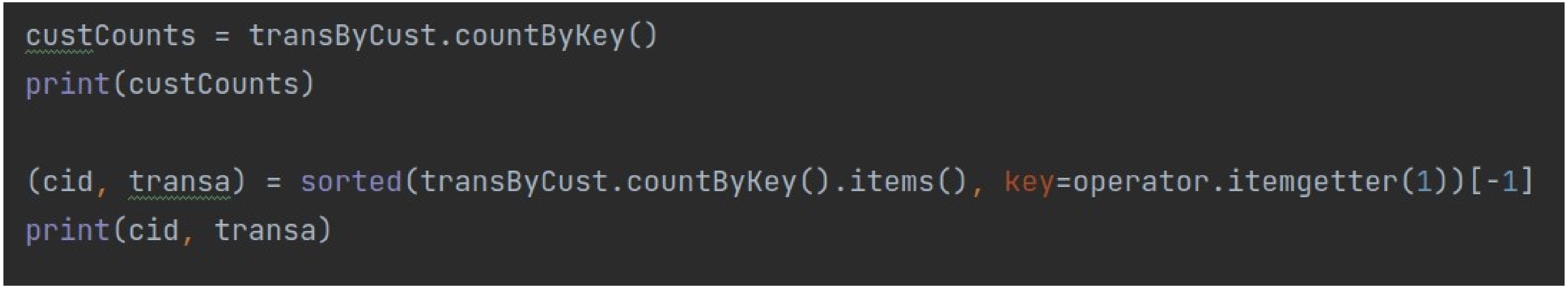
```
2015-03-30#6:55 AM#51#68#1#9506.21
2015-03-30#7:39 PM#99#86#5#4107.59
2015-03-30#11:57 AM#79#58#7#2987.22
2015-03-30#12:46 AM#51#50#6#7501.89
2015-03-30#11:39 AM#86#24#5#8370.2
2015-03-30#10:35 AM#63#19#5#1023.57
2015-03-30#2:30 AM#23#77#7#5892.41
2015-03-30#7:41 PM#49#58#4#9298.18
2015-03-30#9:18 AM#97#86#8#9462.89
2015-03-30#10:06 PM#94#26#4#4199.15
2015-03-30#10:57 AM#91#18#1#3795.73
2015-03-30#7:43 AM#20#86#10#1477.35
2015-03-30#5:58 PM#38#39#6#1090.0
2015-03-30#1:08 PM#46#6#10#1014.78
2015-03-30#12:18 AM#56#48#9#8346.42
2015-03-30#1:18 AM#11#58#4#364.59
2015-03-30#3:01 AM#59#9#5#5984.68
2015-03-30#11:44 AM#8#35#6#1859.2
```

Pair RDD

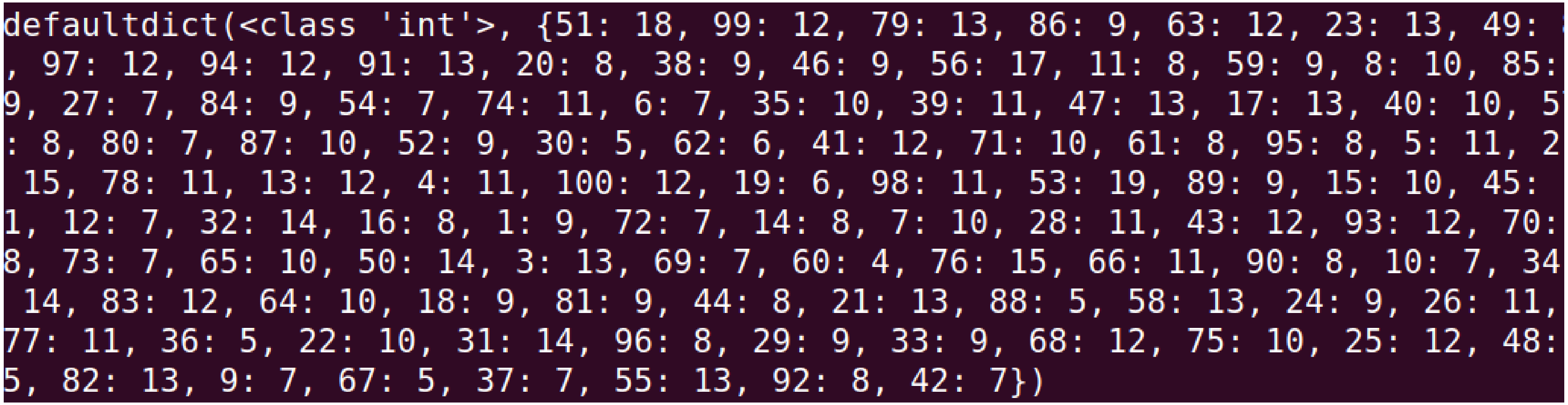
- RDD sa (key, value) elementima
- 

```
if __name__ == '__main__':
    sc = SparkContext('local')
    spark = SparkSession(sc)

    tranFile = sc.textFile("./ch04_data_transactions.txt")
    transData = tranFile.map(lambda transa: transa.split("#"))
    transByCust = transData.map(lambda transa: (int(transa[2]), transa))
    print(transByCust.keys().distinct().count())
    print(transByCust.collect())
```

countByKeys

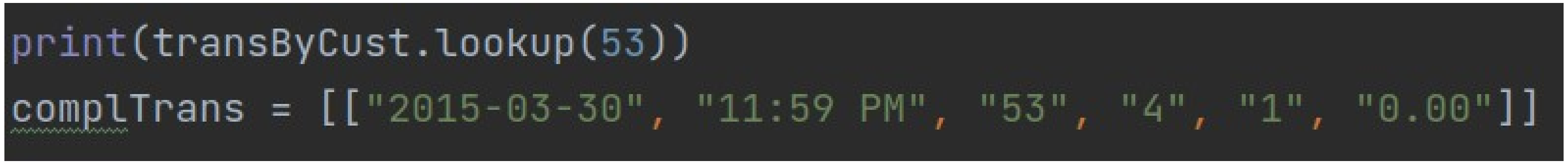


```python
custCounts = transByCust.countByKey()
print(custCounts)

(cid, transa) = sorted(transByCust.countByKey().items(), key=operator.itemgetter(1))[-1]
print(cid, transa)
```

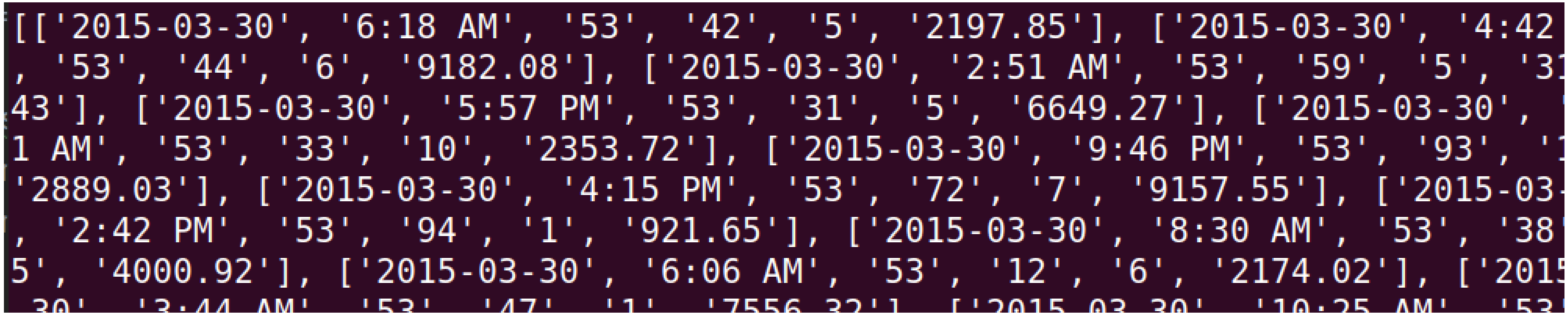


```
defaultdict(<class 'int'>, {51: 18, 99: 12, 79: 13, 86: 9, 63: 12, 23: 13, 49:
, 97: 12, 94: 12, 91: 13, 20: 8, 38: 9, 46: 9, 56: 17, 11: 8, 59: 9, 8: 10, 85:
9, 27: 7, 84: 9, 54: 7, 74: 11, 6: 7, 35: 10, 39: 11, 47: 13, 17: 13, 40: 10, 5
: 8, 80: 7, 87: 10, 52: 9, 30: 5, 62: 6, 41: 12, 71: 10, 61: 8, 95: 8, 5: 11, 2
 15, 78: 11, 13: 12, 4: 11, 100: 12, 19: 6, 98: 11, 53: 19, 89: 9, 15: 10, 45:
1, 12: 7, 32: 14, 16: 8, 1: 9, 72: 7, 14: 8, 7: 10, 28: 11, 43: 12, 93: 12, 70:
8, 73: 7, 65: 10, 50: 14, 3: 13, 69: 7, 60: 4, 76: 15, 66: 11, 90: 8, 10: 7, 34
 14, 83: 12, 64: 10, 18: 9, 81: 9, 44: 8, 21: 13, 88: 5, 58: 13, 24: 9, 26: 11,
77: 11, 36: 5, 22: 10, 31: 14, 96: 8, 29: 9, 33: 9, 68: 12, 75: 10, 25: 12, 48:
5, 82: 13, 9: 7, 67: 5, 37: 7, 55: 13, 92: 8, 42: 7})
```

lookup

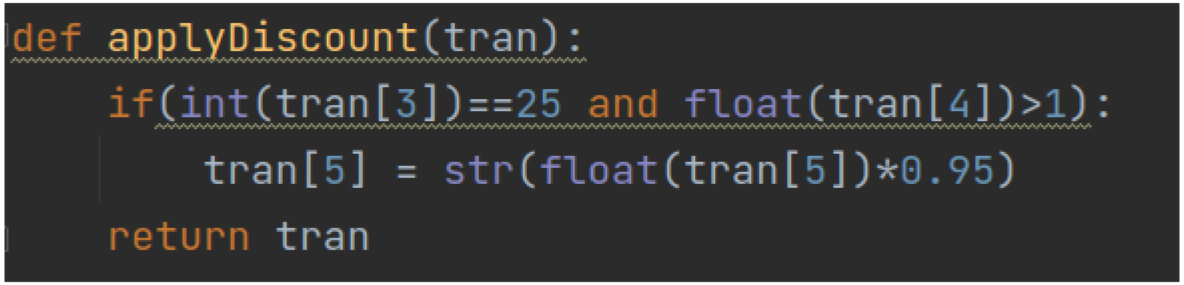


```
print(transByCust.lookup(53))
complTrans = [["2015-03-30", "11:59 PM", "53", "4", "1", "0.00"]]
```



```
[['2015-03-30', '6:18 AM', '53', '42', '5', '2197.85'], ['2015-03-30', '4:42
, '53', '44', '6', '9182.08'], ['2015-03-30', '2:51 AM', '53', '59', '5', '31
43'], ['2015-03-30', '5:57 PM', '53', '31', '5', '6649.27'], ['2015-03-30', '
1 AM', '53', '33', '10', '2353.72'], ['2015-03-30', '9:46 PM', '53', '93', '1
'2889.03'], ['2015-03-30', '4:15 PM', '53', '72', '7', '9157.55'], ['2015-03-
, '2:42 PM', '53', '94', '1', '921.65'], ['2015-03-30', '8:30 AM', '53', '38'
5', '4000.92'], ['2015-03-30', '6:06 AM', '53', '12', '6', '2174.02'], ['2015
-30', '3:44 AM', '53', '47', '1', '7556.32'], ['2015-03-30', '10:25 AM', '53'
```

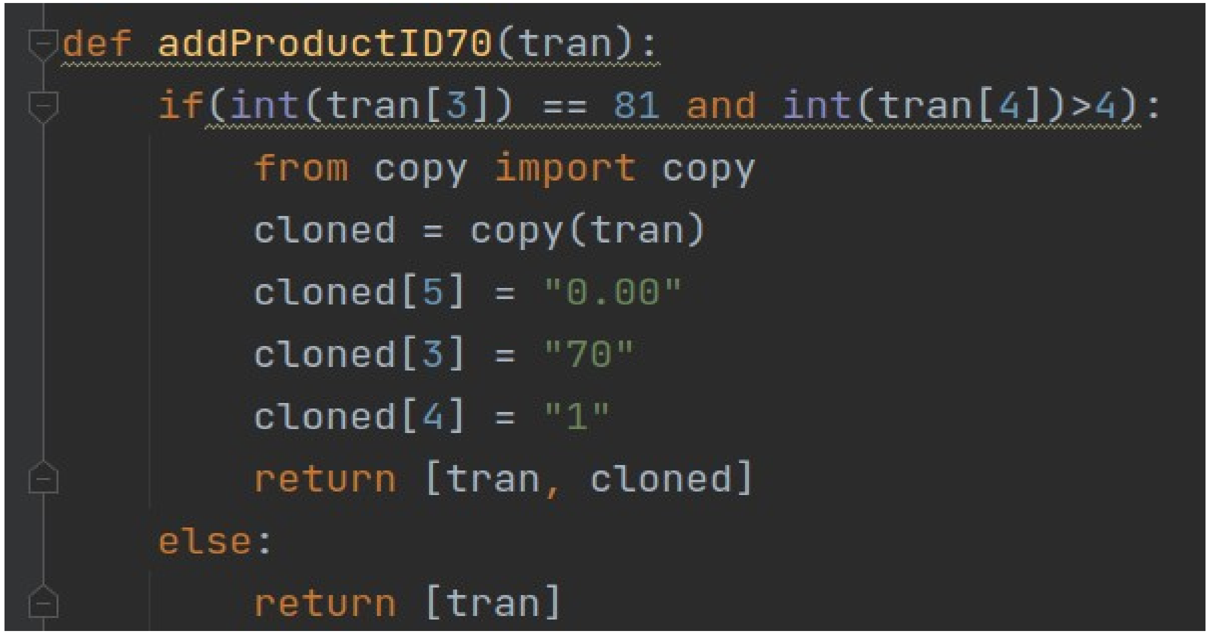
mapValues



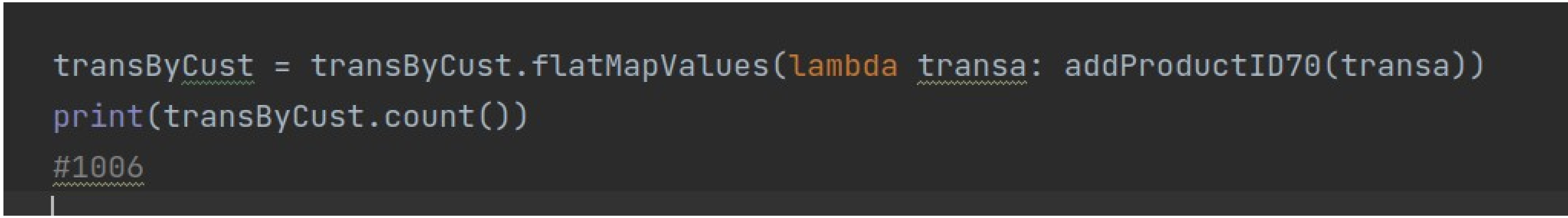
```
def applyDiscount(tran):
    if(int(tran[3])==25 and float(tran[4])>1):
        tran[5] = str(float(tran[5])*0.95)
    return tran
```

transByCust = transByCust.mapValues(lambda t: applyDiscount(t))

flatMapValues

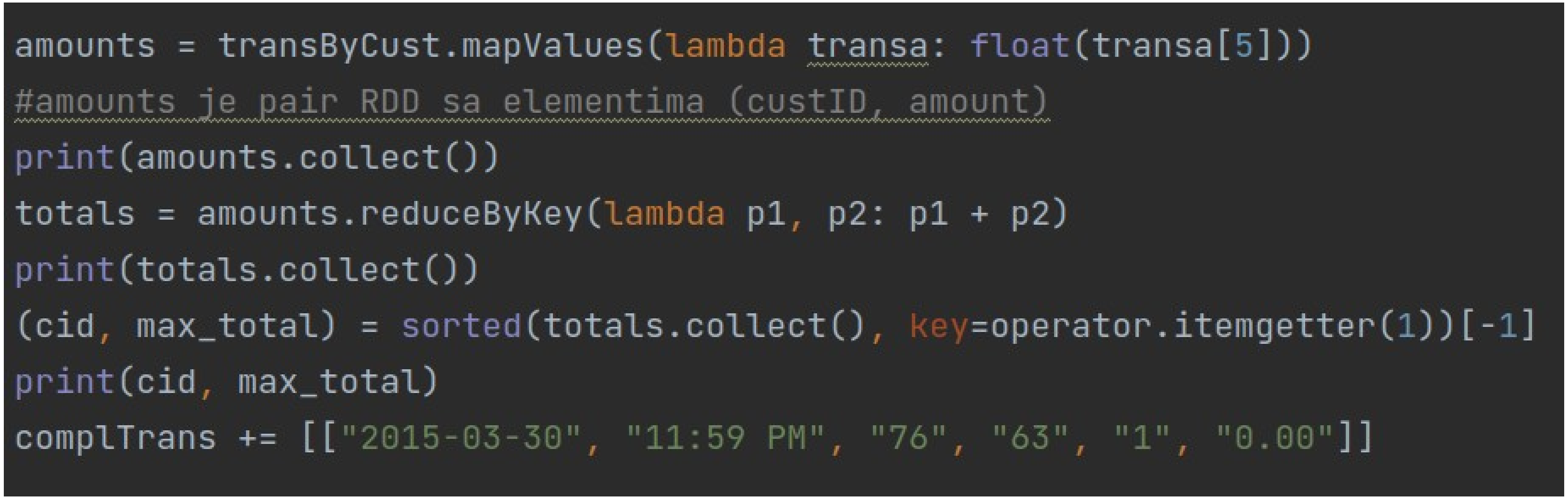


```
def addProductID70(tran):
    if(int(tran[3]) == 81 and int(tran[4])>4):
        from copy import copy
        cloned = copy(tran)
        cloned[5] = "0.00"
        cloned[3] = "70"
        cloned[4] = "1"
        return [tran, cloned]
    else:
        return [tran]
```



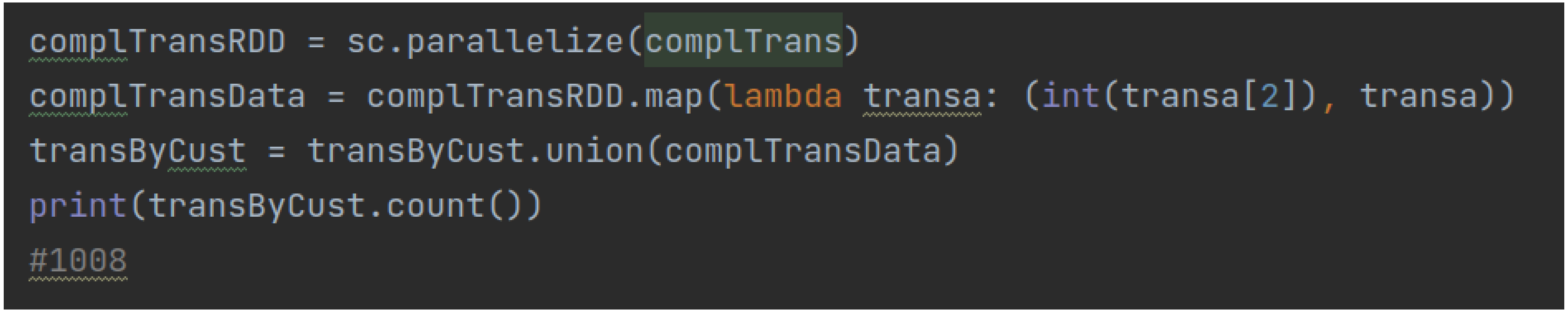
```
transByCust = transByCust.flatMapValues(lambda transa: addProductID70(transa))
print(transByCust.count())
#1006
```

reduceByKey



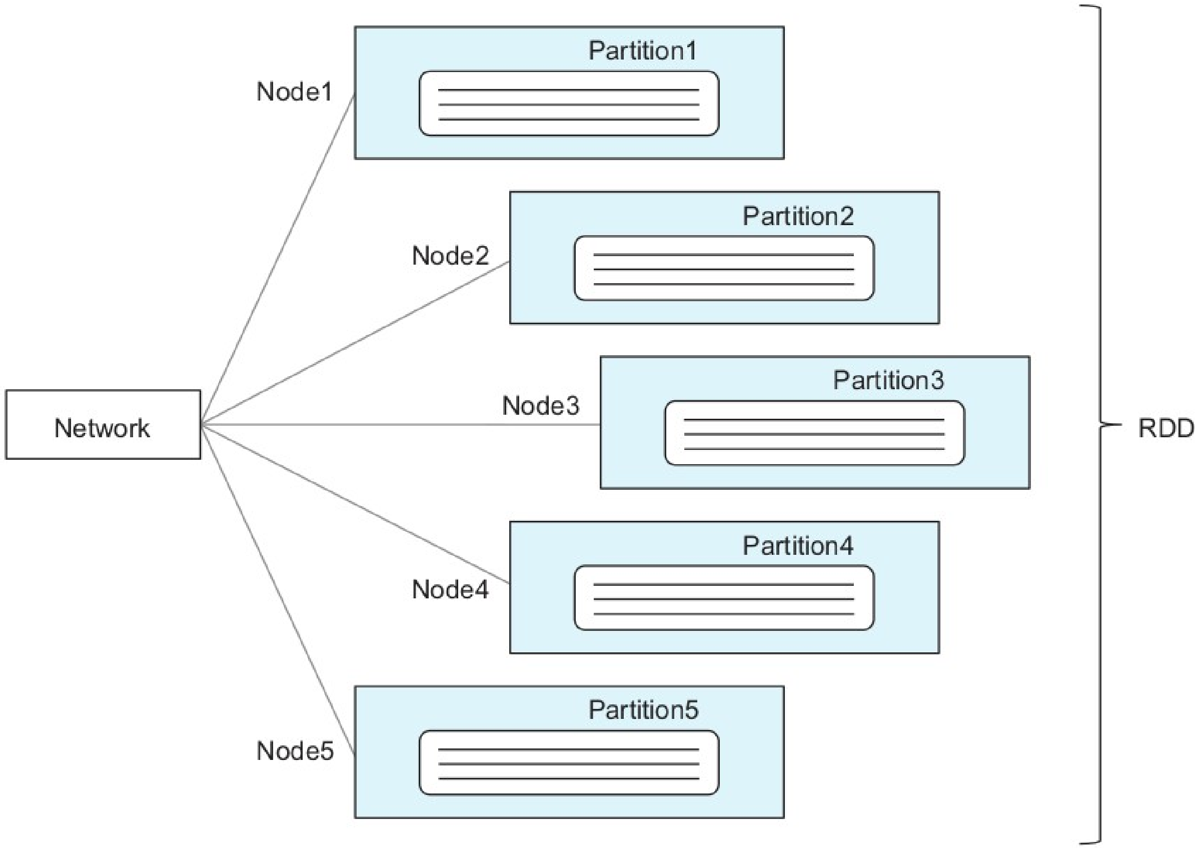
```
amounts = transByCust.mapValues(lambda transa: float(transa[5]))
#amounts je pair RDD sa elementima (custID, amount)
print(amounts.collect())
totals = amounts.reduceByKey(lambda p1, p2: p1 + p2)
print(totals.collect())
(cid, max_total) = sorted(totals.collect(), key=operator.itemgetter(1))[-1]
print(cid, max_total)
complTrans += [["2015-03-30", "11:59 PM", "76", "63", "1", "0.00"]]
```

parallelize, union

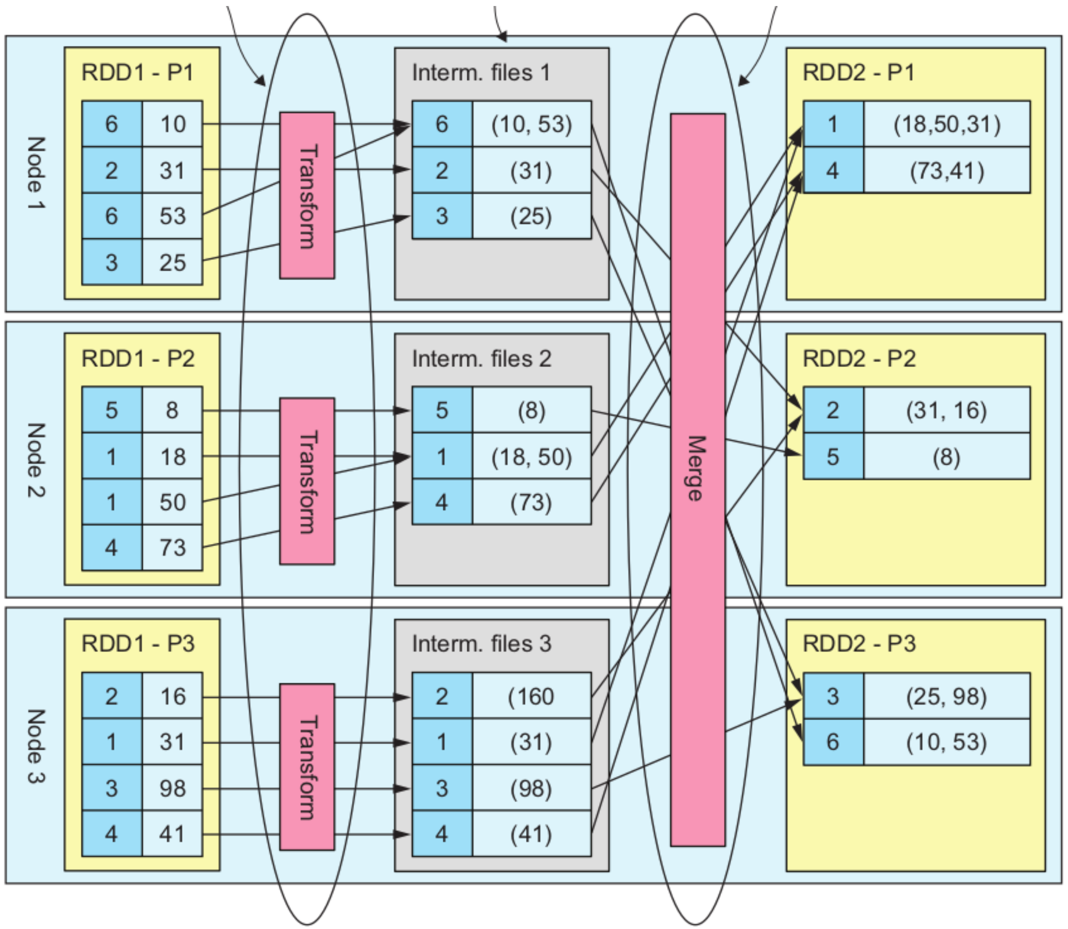


```
complTransRDD = sc.parallelize(complTrans)
complTransData = complTransRDD.map(lambda transa: (int(transa[2]), transa))
transByCust = transByCust.union(complTransData)
print(transByCust.count())
#1008
```

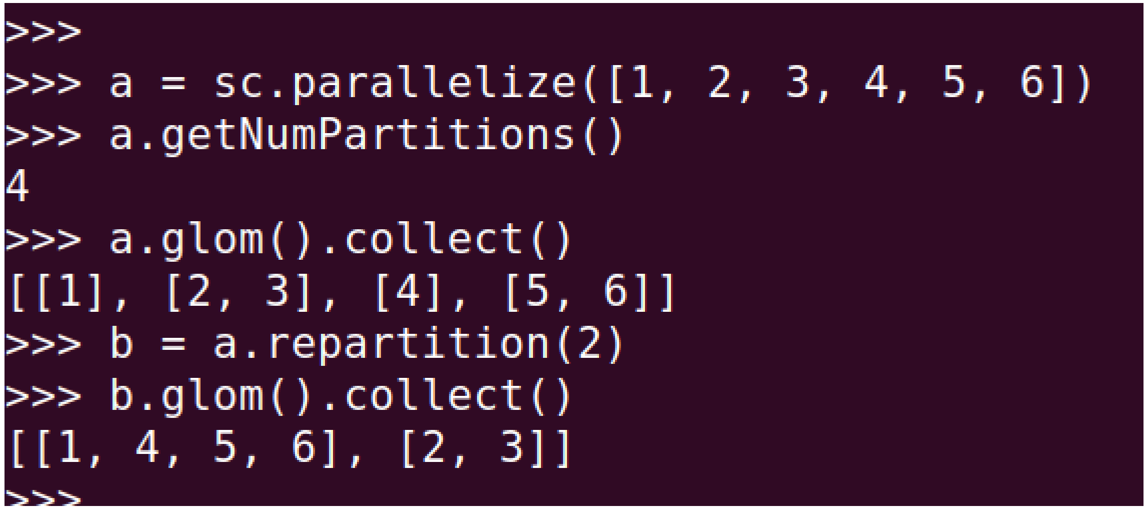
Data partitioning



Data shuffling



Primjer



```
>>>
>>> a = sc.parallelize([1, 2, 3, 4, 5, 6])
>>> a.getNumPartitions()
4
>>> a.glom().collect()
[[1], [2, 3], [4], [5, 6]]
>>> b = a.repartition(2)
>>> b.glom().collect()
[[1, 4, 5, 6], [2, 3]]
>>>
```

Data shuffling 2

- Operacije koje uzrokuju shuffling
  - Pair RDD transformacije: aggregateByKey, foldByKey, reduceByKey, groupByKey, join, leftOuterJoin, rightOuterJoin, fullOuterJoin, subtractByKey
  - RDD transformacije: subtract, intersection, groupWith
  - sortByKey
  - partitionBy ili coalesce sa parametrom shuffle=true