Homework8

Problem 1: Find Low Cost Sites (25%)

(a)

1) Using Pig:
   After loading all the data, call: sampleData = LIMIT data 10;
2) Not using Pig:
   Unix command: head -n 10 ~/training_materials/analyst/data/ad_data1.txt > sample.txt
   And then load the data in "sampleData".
   Testing local saves the time to communicate with HDFS. Also, testing on a small sample of data saves the time to process the whole dataset.

(b) hadoop fs -cat /dualcore/ad_data1.txt | head -100 > test_ad_data.txt

(c)

```
(diskcentral.example.com,68)
(megawave.example.com,96)
(megasource.example.com,100)
(salestiger.example.com,141)
```

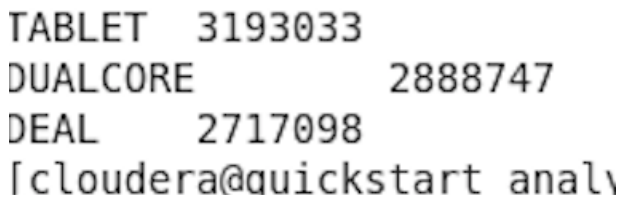
(d)

```
bassoonenthusiast.example.com   1246
grillingtips.example.com        4800
footwear.example.com    4898
coffeenews.example.com  5106
```

Problem 2: Find High Cost Keywords (10%)

(a) See committed code.

(b)



Problem 3: Calculate Click-Through Rate (15%)

(a) See committed code.

(b)

Problem 4: ETL with SPARK (40%)

(a) We can use wholeTextFiles() to create an RDD from the activations dataset. Each file in the directory is mapped to a single RDD element. Each element is a tuple in the form of (file-name, file-content). The first value is the name of file and the second value is the content.

Commands:

```
>>> hw8_4a=sc.wholeTextFiles("/loudacre/activations")
```

(b) We can use map(function) to get the information in the second value. Then use flatMap(function) to separate RDD elements. Before operation, we need to define three functions as below.

Commands:

```
>>> import xml.etree.ElementTree as ElementTree
>>> def getactivations(s):
...     filetree = ElementTree.fromstring(s)
...     return filetree.getiterator('activation')
...
>>> def getmodel(activation):
...     return activation.find('model').text
...
>>> def getaccount(activation):
...     return activation.find('account-number').text
...

>>> hw8_4b=hw8_4a.map(lambda l:l[1]).flatMap(lambda m:getactivations(m))
```

(c) Commands:

```
>>> hw8_4c=hw8_4b.map(lambda l:(getaccount(l),getmodel(l))).map(lambda m:m[0]+":"+m[1])

>>> hw8_4c.saveAsTextFile("/loudacre/account_model")
```

The resulting file is below.

File Edit View Search Terminal Help

```
62881:Sorrento F41L
65088:Sorrento F41L
85995:Sorrento F41L
126018:Titanic 2500
27810:Sorrento F41L
16047:Sorrento F41L
2715:Sorrento F41L
124218:Sorrento F31L
129326:Sorrento F00L
23387:Titanic 2000
79605:Sorrento F41L
76819:Sorrento F41L
105961:Sorrento F41L
129637:iFruit 3A
105359:Sorrento F41L
90274:Sorrento F41L
81952:MeeToo 3.1
43370:Sorrento F41L
38790:Sorrento F41L
119481:Sorrento F41L
115708:Sorrento F41L
127965:iFruit 4
127677:iFruit 1
83750:Titanic 3000
4879:Sorrento F41L
78899:Sorrento F41L
38955:Sorrento F41L
127006:Titanic 2400
2607:Sorrento F41L
122571:iFruit 4A
108631:Sorrento F41L
23004:Sorrento F41L
33801:Sorrento F41L
127865:Titanic 2400
114113:Sorrento F41L
80994:Sorrento F41L
119295:Sorrento F41L
102514:Sorrento F41L
30920:Sorrento F41L
70904:Sorrento F41L
128222:iFruit 4
```

Problem 5: SPARK Job Execution (10%)

(a) Pipelining means that Spark will perform sequences of transformations by row when it is possible so no data is stored. Operations are only performed on the data records that are necessary to produce the return value.
The benefit is that intermediate records or RDDs do not need to be stored.

(b) Example of two operations that can be pipelined together: map() and filter()

```
val mydata_uc = mydata.map(line =>line.toUpperCase())
val mydata_filt = mydata_uc.filter(line=> line.startsWith("I"))
```

(c) Example of two operations that cannot be pipelined together: first() and take(2).
Because the intermediate records have to be recorded.