BDA1 - Spark

Martynas Lukosevicius, Alejo Perez Gomez

24/04/2021

1.

What are the lowest and highest temperatures measured each year for the period 1950 - 2014. Provide the lists sorted in the descending order with respect to the maximum temperature. In this exercise you will use the temperature-readings.csv file. The output should at least contain the following information (You can also include a Station column so that you may find multiple stations that record the highest (lowest) temperature.):

```
from pyspark import SparkContext

sc = SparkContext(appName = "exercise 1")
# This path is to the file on hdfs
temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
lines = temperature_file.map(lambda line: line.split(";"))

# (key, value) = (year,temperature)
year_temperature = lines.map(lambda x: (x[1][0:4], float(x[3])))

#filter
year_temperature = year_temperature.filter(lambda x: int(x[0])>=1950 and int(x[0])<=2014)

#Get max
max_temperatures = year_temperature.reduceByKey(lambda a,b:  max(a, b))

max_temperatures = max_temperatures.sortBy(ascending = False, keyfunc=lambda k: k[1])

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder
max_temperatures.saveAsTextFile("BDA/output")
```

```
from pyspark import SparkContext

sc = SparkContext(appName = "exercise 1")
# This path is to the file on hdfs
temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
lines = temperature_file.map(lambda line: line.split(";"))

# (key, value) = (year,temperature)
year_temperature = lines.map(lambda x: (x[1][0:4], float(x[3])))

#filter
year_temperature = year_temperature.filter(lambda x: int(x[0])>=1950 and int(x[0])<=2014)

#Get min
```

```
min_temperatures = year_temperature.reduceByKey(lambda a,b:  min(a, b))

min_temperatures = min_temperatures.sortBy(ascending = False, keyfunc=lambda k: k[1])


# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder
min_temperatures.saveAsTextFile("BDA/output")
```

Results:

Max

(u'1975', 36.1)

(u'1992', 35.4)

(u'1994', 34.7)

(u'2014', 34.4)

(u'2010', 34.4)

(u'1989', 33.9)

(u'1982', 33.8)

(u'1968', 33.7)

(u'1966', 33.5)

(u'1983', 33.3)

Min

(u'1990', -35.0)

(u'1952', -35.5)

(u'1974', -35.6)

(u'1954', -36.0)

(u'1992', -36.1)

(u'1975', -37.0)

(u'1972', -37.5)

(u'1995', -37.6)

(u'2000', -37.6)

(u'1957', -37.8)

2)

Count the number of readings for each month in the period of 1950-2014 which are higher than10degrees.Repeattheexercise,thistimetakingonlydistinctreadingsfromeachstation. That is, if a station reported a reading above 10 degrees in some month, then itappears only once in the count for that month.

```
from pyspark import SparkContext

sc = SparkContext(appName = "exercise 1")
# This path is to the file on hdfs
```

```
temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
lines = temperature_file.map(lambda line: line.split(";"))

# (key, value) = (year,temperature)
year_temperature = lines.map(lambda x: (x[1][0:7], (x[0] ,float(x[3]))))

#filter
year_temperature = year_temperature.filter(lambda x: int(x[0][0:4])>=1950 and int(x[0][0:4])<=2014)
year_temperature = year_temperature.filter(lambda x: x[1][1]>=10).map(lambda x: (x[0], 1))

#Get max
max_temperature = year_temperature.reduceByKey(lambda a,b: a+b)
max_temperature = max_temperature.sortBy(ascending = False, keyfunc=lambda k: k[0])

#print(max_temperatures.collect())

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder
max_temperature.saveAsTextFile("BDA/output")
```

Results:

```
(u'2014-12', 5)

(u'2014-11', 8498)

(u'2014-10', 43359)

(u'2014-09', 87131)

(u'2014-08', 125006)

(u'2014-07', 147910)
```

```
from pyspark import SparkContext

sc = SparkContext(appName = "exercise 1")
# This path is to the file on hdfs
temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
lines = temperature_file.map(lambda line: line.split(";"))

# (key, value) = (year,temperature)
year_temperature = lines.map(lambda x: ((x[1][0:4], x[1][5:7],x[0] ), float(x[3])))

#filter
year_temperature = year_temperature.filter(lambda x: int(x[0][0])>=1950 and int(x[0][0])<=2014)
year_temperature = year_temperature.filter(lambda x: x[1]>=10).map(lambda x: x[0]).distinct()

year_temperature = year_temperature.map(lambda x: ((x[0],x[1]),1))

#Get max
max_temperature = year_temperature.reduceByKey(lambda a,b: a+b)
max_temperature = max_temperature.sortBy(ascending = False, keyfunc=lambda k: k[0])

#print(max_temperatures.collect())

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder
max_temperature.saveAsTextFile("BDA/output")
```

Results:

((u'2014', u'12'), 3)

((u'2014', u'11'), 160)

((u'2014', u'10'), 272)

((u'2014', u'09'), 296)

((u'2014', u'08'), 296)

((u'2014', u'07'), 297)

((u'2014', u'06'), 298)

((u'2014', u'05'), 296)

((u'2014', u'04'), 254)

3)

Find the average monthly temperature for each available station in Sweden. Your result should include average temperature for each station for each month in the period of 1960- 2014.

```
from pyspark import SparkContext

sc = SparkContext(appName = "exercise 1")
# This path is to the file on hdfs
temperature_file = sc.textFile("C:\\Users\\marty\\Desktop\\temperature-readings.csv")
lines = temperature_file.map(lambda line: line.split(";"))

# (key, value) = (year,temperature)
year_temperature = lines.map(lambda x: ((x[1][0:4], x[1][5:7],x[1][8:10], x[0]), (float(x[3]), float(x[3

#filter
year_temperature = year_temperature.filter(lambda x: int(x[0][0])>=1960 and int(x[0][0])<=2014)


#Get max
max_temperature = year_temperature.reduceByKey(lambda a,b: (min(a[0],b[0]), max(a[1],b[1]))).map(lambda
max_temperature = max_temperature.reduceByKey(lambda a,b: (a[0]+b[0],a[1]+b[1],a[2]+b[2])).map(lambda x
max_temperature = max_temperature.sortBy(ascending = False, keyfunc=lambda k: k[1])


# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder

max_temperature.saveAsTextFile("BDA/output")
```

Results:

((u'2014', u'07', u'96000'), 26.3)

((u'1994', u'07', u'96550'), 23.071052631578947)

((u'1983', u'08', u'54550'), 23.0)

((u'1994', u'07', u'78140'), 22.970967741935482)

((u'1994', u'07', u'85280'), 22.87258064516129)

4)

Provide a list of stations with their associated maximum measured temperatures and maximum measured daily precipitation. Show only those stations where the maximum temperature is between 25 and 30 degrees and maximum daily precipitation is between 100 mm and 200mm.

```
from pyspark import SparkContext

sc = SparkContext(appName = "exercise 1")
# This path is to the file on hdfs
temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
prec_file = sc.textFile("BDA/input/precipitation-readings.csv")
lines_temp = temperature_file.map(lambda line: line.split(";"))
lines_prec = prec_file.map(lambda line: line.split(";"))

# (key, value) = (year,temperature)
year_temperature = lines_temp.map(lambda x: ( x[0] , float(x[3]))).reduceByKey(lambda a,b: max(a,b))
prec_measures = lines_prec.map(lambda x: (x[0], float(x[3]))).reduceByKey(lambda a,b: max(a,b))

final =  year_temperature.join(prec_measures)

final = final.filter(lambda x: float(x[1][0])>=25 and float(x[1][0])<=30).filter(lambda x: float(x[1][1

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder
final.saveAsTextFile("BDA/output")
```

Result:

Empty

5)

Calculate the average monthly precipitation for the Östergotland region (list of stations is provided in the separate file) for the period 1993-2016. In order to do this, you will first need to calculate the total monthly precipitation for each station before calculating the monthly average (by averaging over stations).

```
from pyspark import SparkContext
import logging

sc = SparkContext(appName = "exercise 1")
# This path is to the file on hdfs

stations = sc.textFile("BDA/input/stations-Ostergotland.csv")
prec_file = sc.textFile("BDA/input/precipitation-readings.csv")

stations = stations.map(lambda line: line.split(";")[0]).collect()
prec = prec_file.map(lambda line: line.split(";")).map(lambda x: (x[0], x[1][0:4], x[1][5:7], x[1][8:10
prec = prec.filter(lambda x: x[0] in stations)
prec = prec.map(lambda x: ((x[0], x[1], x[2]), x[4])).reduceByKey(lambda a,b: a + b).map(lambda x: ((x[

prec = prec.reduceByKey(lambda a,b: (a[0]+b[0],a[1]+b[1]))

prec = prec.mapValues(lambda x: (x[0]/x[1])).sortBy(ascending = False, keyfunc=lambda k: k[1])

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder
```

```
prec.saveAsTextFile("BDA/output")
```

Result:

```
((u'2006', u'08'), 148.0833333333333)
((u'2008', u'08'), 138.51666666666654)
((u'2000', u'07'), 135.8666666666666)
((u'1995', u'09'), 134.55)
((u'2012', u'06'), 132.2)
```