Lab 1 Report

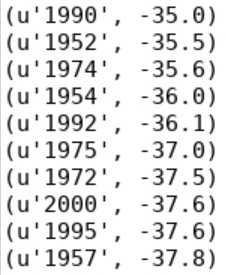
Group member: Hong Zhang(honzh073) Bing Rui Yao(binya518)

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.

Code:

|  |  |
| --- | --- |
| 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: a if a>=b else b) | |
| max\_temperatures = max\_temperatures.sortBy(ascending = False, keyfunc=lambda k: k[1]) | |
| **#Get min** | |
| #min\_temperatures = year\_temperature.reduceByKey(lambda a,b: a if a<=b else b) | |
| #min\_temperatures = min\_temperatures.sortBy(ascending = False, keyfunc=lambda k: k[1]) | |
| print(max\_temperatures.collect()) | |
| **# Following code will save the result into /user/ACCOUNT\_NAME/BDA/output folder** | |
| max\_temperatures.saveAsTextFile("BDA/output") |  |

First 10 rows of lowest temperatures:



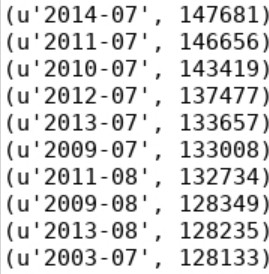
First 10 rows of highest temperatures:



1. Count the number of readings for each month in the period of 1950-2014 which are higher than 10 degrees.

Code:

|  |  |  |
| --- | --- | --- |
| y | 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-month,temperature)** | |
| year\_temperature = lines.map(lambda x: (x[1][0:7],float(x[3]))) | |
| **#filter** | |
| ear\_temperature = year\_temperature.filter(lambda x: int(x[0][0:4])>=1950 and |  |
| int(x[0][0:4])<=2014 and x[1]>10) | | |
|  | **#map** | |
| year\_temperature = year\_temperature.map(lambda x: (x[0],1)) | |
| **#count** | |
| count\_temperatures=year\_temperature.reduceByKey(lambda a,b: a+b) | |
| count\_temperatures=count\_temperatures.sortBy(ascending=False, keyfunc=lambda k: k[1]) | |
| #print(count\_temperatures.collect()) | |
| **# Following code will save the result into /user/ACCOUNT\_NAME/BDA/output folder** | |
| count\_temperatures.saveAsTextFile("BDA/output") | |

First 10 rows of the readings:

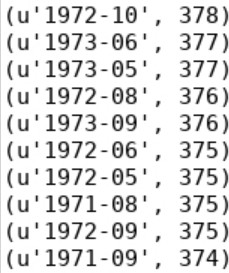
Repeat the exercise, this time taking only distinct readings from each station. That is, if a station reported a reading above 10 degrees in some month, then it appears only once in the count for that month.

Code:

|  |  |
| --- | --- |
| 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-month,station),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][0:4])>=1950 and |  |

|  |  |
| --- | --- |
| int(x[0][0][0:4])<=2014 and x[1]>10) | |
|  | **#count** |
| year\_temperature = year\_temperature.reduceByKey(max) |
| year\_temperature = year\_temperature.map(lambda x: (x[0][0],1)) |
| count\_temperatures=year\_temperature.reduceByKey(lambda a,b: a+b) |
| count\_temperatures=count\_temperatures.sortBy(ascending = False, keyfunc=lambda k: k[1]) |
| #print(max\_temperatures.collect()) |
| **# Following code will save the result into /user/ACCOUNT\_NAME/BDA/output folder** |
| count\_temperatures.saveAsTextFile("BDA/output") |

First 10 rows of the distinct readings:



1. 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.

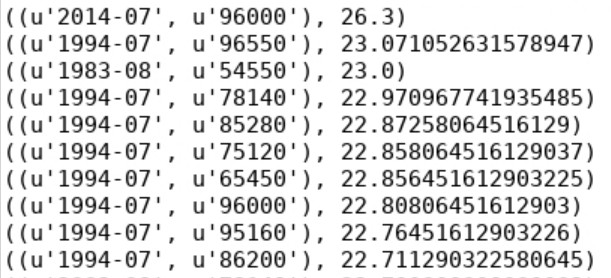
Code:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 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-month-date,station),temperature)** | | | | |
| date\_temperature = lines.map(lambda x: ((x[1][0:10],x[0]),float(x[3]))) | | | | |
| **#filter** | | | | |
| date\_temperature = date\_temperature.filter(lambda x: int(x[0][0][0:4])>=1960 and | | |  | |
| int(x[0][0][0:4])<=2014) | | | | | |
|  | **#Get max and min** | | | | |
| **#(key, value) = ((year-month-date,station),(max,min))** | | | | |
| max\_temperatures = date\_temperature.reduceByKey(max) | | | | |
| min\_temperatures = date\_temperature.reduceByKey(min) | | | | |
| maxmin\_temperatures = max\_temperatures.join(min\_temperatures) | | | | |
| **#reduce** | | | | |
| **#(key, value) = ((year-month,station),(max,min,1))** | | | | |
| month\_temperature = maxmin\_temperatures.map(lambda | |  | | |
| :((x[0][0][0:7],x[0][1]),(x[1][0],x[1][1],int(1)))) | | | | | |
|  | ave\_temperature = month\_temperature.reduceByKey(lambda a,b: (a[0]+b[0],a[1]+b[1],a[2]+b[2])) | | | |  |
| ave\_temperature = ave\_temperature.map(lambda |  | | | |

x

|  |  |
| --- | --- |
| x:(x[0],(x[1][0]+x[1][1])/(x[1][2]\*2))).sortBy(ascending = False, keyfunc=lambda k: k[1]) | |
|  | #print(ave\_temperature.collect()) |
| # Following code will save the result into /user/ACCOUNT\_NAME/BDA/output folder |
| ave\_temperature.saveAsTextFile("BDA/output") |

First 10 rows of the average monthly temperatures:

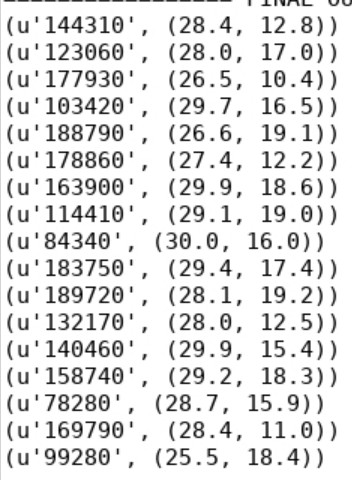


1. Provide a list of stations with their associated maximum measured temperatures and maximum measured daily precipitation. Show only those stations where the maximum temperature is between 25 and 30 degrees and maximum daily precipitation is between 100 mm and 200mm.

Code:

|  |
| --- |
| 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") |
| tem\_lines = temperature\_file.map(lambda line: line.split(";")) |
| precipitation\_file = sc.textFile("BDA/input/precipitation-readings.csv") |
| pre\_lines = precipitation\_file.map(lambda line: line.split(";")) |
| **# (key, value) = (station,temperature)** |
| station\_temp = tem\_lines.map(lambda x: (x[0],float(x[3]))) |
| **# (key, value) = (station,precipitation)** |
| station\_pre = pre\_lines.map(lambda x: (x[0],float(x[3]))) |
| **#Get max, filter** |
| max\_temp = station\_temp.reduceByKey(lambda a,b: a if a>=b else b) |
| max\_temp = max\_temp.filter(lambda x: x[1]>=25 and x[1]<=30) |
| max\_pre = station\_pre.reduceByKey(lambda a,b: a if a>=b else b) |
| max\_pre = max\_pre.filter(lambda x: x[1]>=10 and x[1]<=20) |
| **#join** |
| station\_max = max\_temp.join(max\_pre) |
| #print(station\_max.collect()) |
| # Following code will save the result into /user/ACCOUNT\_NAME/BDA/output folder |
| station\_max.saveAsTextFile("BDA/output") |

Output:



1. 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).

Code:

|  |  |  |  |
| --- | --- | --- | --- |
|  | from pyspark import SparkContext | | |
| sc = SparkContext(appName = "exercise 1") | | |
| # This path is to the file on hdfs | | |
| pre\_file = sc.textFile("BDA/input/precipitation-readings.csv") | | |
| pre\_lines = pre\_file.map(lambda line: line.split(";")) | | |
| o\_file = sc.textFile("BDA/input/stations-Ostergotland.csv") | | |
| o\_lines = o\_file.map(lambda line: line.split(";")) | | |
| station\_list = o\_lines.map(lambda x: x[0]).collect() | | |
| b\_station\_list = sc.broadcast(station\_list) | | |
| # (key, value) = ((year-month,station),precipitation) | | |
| pre\_all = pre\_lines.map(lambda x: ((x[1][0:7],x[0]),float(x[3]))) | | |
| pre\_o = pre\_all.filter(lambda x: int(x[0][0][0:4])>=1993 and int(x[0][0][0:4])<=2016 and x[0][1] in | |  |
| b\_station\_list.value) | | | |
|  | #Get monthly precipitation | | |
| # (key,value)=((year-month,station),month\_pre) | | |
| pre\_o\_month = pre\_o.reduceByKey(lambda a,b: a+b) | | |
| # (key,value)=((year-month),(month\_pre,1)) | | |
| pre\_o\_monthavg = pre\_o\_month.map(lambda x: (x[0][0],(x[1],1))) | | |
| pre\_o\_monthavg = pre\_o\_monthavg.reduceByKey(lambda a,b: (a[0]+b[0],a[1]+b[1])) | | |
| pre\_o = pre\_o\_monthavg.map(lambda x : (x[0],x[1][0]/x[1][1])).sortBy(ascending = False, |  | |
| keyfunc=lambda k: k[1]) | | | |
|  | print(pre\_o.collect()) | | |
| # Following code will save the result into /user/ACCOUNT\_NAME/BDA/output folder | | |
| pre\_o.saveAsTextFile("BDA/output") | | |

First 10 rows of output:

