732A54 Big Data Analytics

Lab 1; pyspark

Group G6:

Hoda Fakharzadehjahromy(hodfa840)
Seyda Aqsa Iftekhar(syeif776)

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.):

Year, temperature

Code:

```
from pyspark import SparkContext
import os
import sys

os.environ['PYSPARK_PYTHON'] = sys.executable
os.environ['PYSPARK_DRIVER_PYTHON'] = sys.executable
sc = SparkContext(appName="ex1" )
# MAP
temperature1 = sc.textFile("data/temperature-readings.csv")
lines = temperature1.map(lambda x: x.split(";"))
#print(lines.take(3))
# filter
year_temperature = lines.map(lambda x: (x[1][0:4],float(x[3]) ))
year_temperature = year_temperature.filter(lambda x: (int(x[0] )>= 1950 and
int(x[0]) <= 2014))
 #reduce
max_temperature = year_temperature.reduceByKey(lambda a,b: a if a >= b else b  )
max_sorted_temperature = max_temperature.sortBy(ascending=False,keyfunc=lambda
k:k[1])

min_temperature = year_temperature.reduceByKey(lambda a,b: a if a < b else b  )

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

#print(min_sorted_temperature.take( 10))
#print("-----------------------------------------------------")
#print(max_sorted_temperature.take( 10))
min_sorted_temperature.saveAsTextFile("res/ex1_min")
max_sorted_temperature.saveAsTextFile("res/ex1_max")
```

Output:

Max:

```
[('1992', 35.4), ('1994', 34.7), ('2014', 34.4), ('2010', 34.4), ('1989', 33.9),
('1982', 33.8), ('1968', 33.7), ('1966', 33.5), ('1983', 33.3), ('2002', 33.3),
('1986', 33.2), ('1970', 33.2), ('1956', 33.0), ('2000', 33.0), ('1959', 32.8),
('2006', 32.7), ('1991', 32.7), ('1988', 32.6), ('2011', 32.5)]
```

Min:

```
[('1999', -49.0), ('1978', -47.7), ('1987', -47.3), ('1967', -45.4), ('1956', -
45.0), ('1980', -45.0), ('1971', -44.3), ('1986', -44.2), ('2001', -44.0),
('1965', -44.0), ('1981', -44.0), ('1979', -44.0), ('1959', -43.6), ('1985', -
43.4), ('1958', -43.0), ('1998', -42.7), ('2012', -42.7), ('2014', -42.5),
('1977', -42.5)]
```

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

In this exercise you will use the *temperature-readings.csv* file.

The output should contain the following information:

Year, month, count

Code: 2a)

```
from pyspark import SparkContext
from operator import add
import os
import sys

os.environ['PYSPARK_PYTHON'] = sys.executable
os.environ['PYSPARK_DRIVER_PYTHON'] = sys.executable

sc = SparkContext(appName="ex2")

temperature= sc.textFile("data/temperature-readings.csv")

lines = temperature.map(lambda line: line.split(";"))
```

```
yearMontemp = lines.map(lambda x: ((x[1][0:4],x[1][5:7]), (x[0], float(x[3]))))
#print(year_month.take(5))
#print("----------------------------------------")
#filtering
yearMontemp = yearMontemp.filter(lambda x: int(x[0][0])>=1950 and
int(x[0][0])<=2014 and x[1][1]>10)
#print(year_month.take(5))
#print("----------------------------------------")
count_yearMontemp= yearMontemp.map(lambda x:(x[0],1)).reduceByKey(add).\
sortByKey(ascending=False)

count_yearMontemp.saveAsTextFile("output2/ex2a")
```

Output 2a):

```
(('2003', '09'), 70459)
(('2003', '08'), 108501)
(('2003', '07'), 128133)
(('2003', '06'), 99693)
(('2003', '05'), 48264)
(('2003', '04'), 11786)
(('2003', '03'), 3581)
(('2003', '02'), 3)
(('2002', '10'), 4939)
(('2002', '09'), 65928)
```

Code 2b) distinct reading for each station:

```
from pyspark import SparkContext
import os
import sys

os.environ['PYSPARK_PYTHON'] = sys.executable
os.environ['PYSPARK_DRIVER_PYTHON'] = sys.executable

sc = SparkContext(appName="ex2")
```

```
temperature= sc.textFile("data/temperature-readings.csv")

lines = temperature.map(lambda line: line.split(";"))
yearMontemp= lines.map(lambda x: ( (x[1][0:4],x[1][5:7]),float(x[3]) ))
yearMontemp =yearMontemp.filter(lambda x: int(x[0][0]) >= 1950 and int(x[0][0])
<= 2014)
yearMontemp=yearMontemp.filter(lambda x: x[1] >10)
#print(yearMontemp.take(5))

yearMontemp=yearMontemp.map(lambda x: (x, 1)).groupByKey().mapValues(len)
yearMontemp.saveAsTextFile("output2/ex2b")
#print(yearMontemp.take(5))
```

Output:

```
((('2014', '06'), 21.1), 227)
((('2014', '07'), 25.0), 615)
((('2014', '07'), 25.5), 583)
((('1955', '10'), 16.7), 1)
((('1955', '10'), 16.4), 17)
((('1956', '07'), 11.4), 189)
((('1956', '07'), 11.8), 216)
((('1956', '07'), 11.0), 210)
((('1956', '07'), 11.6), 145)
((('1956', '07'), 11.2), 145)
((('1956', '07'), 11.5), 31)
((('1958', '08'), 13.8), 599)
((('1958', '08'), 13.0), 819)
((('1958', '08'), 13.6), 561)
((('1958', '08'), 13.2), 573)
((('1958', '08'), 13.4), 583)
((('1959', '09'), 10.3), 52)
((('1959', '09'), 10.1), 41)
((('1959', '09'), 10.2), 332)
((('1959', '09'), 10.4), 331)
((('1959', '09'), 10.6), 267)
((('1960', '06'), 16.4), 376)
((('1960', '06'), 16.2), 344)
((('1960', '06'), 16.6), 309)
((('1960', '06'), 16.0), 693)
((('1960', '06'), 16.1), 60)
((('1960', '07'), 20.2), 167)
((('1960', '07'), 20.8), 97)
((('1960', '07'), 20.0), 354)
((('1961', '10'), 10.8), 731)
((('1961', '10'), 10.6), 679)
((('1961', '10'), 10.2), 703)
```

```
((('1961', '10'), 10.5), 289)
((('1961', '10'), 10.7), 169)
((('1961', '10'), 10.4), 767)
((('1961', '10'), 10.3), 210)
((('1963', '08'), 13.2), 732)
((('1963', '08'), 13.0), 1796)
((('1963', '08'), 13.8), 746)
((('1963', '08'), 13.6), 692)
((('1963', '08'), 13.3), 166)
((('1963', '08'), 13.7), 182)
((('1964', '06'), 19.1), 48)
((('1964', '06'), 19.6), 184)
((('1964', '06'), 19.2), 245)
((('1964', '08'), 14.4), 659)
((('1964', '08'), 14.0), 2066)
```

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. Bear in mind that not every station has the readings for each month in this timeframe.

In this exercise you will use the *temperature-readings.csv* file.

The output should contain the following information:

Year, month, station number, average monthly temperature

Code:

```
from pyspark import SparkContext
import os
import sys

os.environ['PYSPARK_PYTHON'] = sys.executable
os.environ['PYSPARK_DRIVER_PYTHON'] = sys.executable

sc = SparkContext(appName='ex3')

temperature= sc.textFile("data/temperature-readings.csv")

lines = temperature.map(lambda line: line.split(";"))

#output format Year, month, station number, average monthly temperature

Year_Mon_temp = lines.map(lambda x :((x[1],x[0]),(float(x[3]))))

daily_average = Year_Mon_temp.groupByKey().mapValues(lambda x: (max(x)+min(x))/2)
```

```
print(daily_average.take(20))

#calculating monthly average
#map as (year, month, station_no), (daily_avg, 1)
monthly_ave = daily_average.map(lambda x:
((x[0][0][0:4],x[0][0][5:7],x[0][1]),(x[1],1))).\
    reduceByKey(lambda x,y: (x[0] + y[0], x[1] + y[1])).map(lambda x: (x[0],
x[1][0]/x[1][1])).sortByKey(False)

print(monthly_ave.take(20))
#
monthly_ave = daily_average.map(lambda x:
((x[0][0][0:4],x[0][0][5:7],x[0][1]),(x[1],1))).
#
monthly_avg= monthly_ave.filter( lambda x: int(x[0][0])>1960 and
int(x[0][0])<2014)

monthly_ave.saveAsTextFile("output3")
```

Output:

```
 (('2013', '12', '102170'), (5.5, 1))
(('2013', '12', '102170'), (5.25, 1))
(('2014', '05', '102170'), (5.2, 1))
(('2014', '05', '102170'), (4.85, 1))
(('2014', '07', '102170'), (20.3, 1))
(('2014', '07', '102170'), (17.2, 1))
(('2014', '12', '102170'), (3.8, 1))
(('2015', '01', '102170'), (-6.5, 1))
(('2015', '04', '102170'), (8.2, 1))
(('2015', '05', '102170'), (6.55, 1))
(('2015', '05', '102170'), (8.95, 1))
(('2015', '11', '102170'), (1.7, 1))
(('2015', '11', '102170'), (-6.8, 1))
(('2016', '01', '102170'), (-2.9, 1))
(('2016', '03', '102170'), (-0.19999999999999996, 1))
(('2016', '04', '102170'), (5.35, 1))
(('1955', '10', '102190'), (7.2, 1))
(('1955', '11', '102190'), (-1.6499999999999995, 1))
(('1956', '01', '102190'), (-8.649999999999999, 1))
(('1956', '06', '102190'), (12.4, 1))
(('1956', '08', '102190'), (13.25, 1))
(('1956', '08', '102190'), (10.75, 1))
(('1956', '08', '102190'), (8.55, 1))
(('1956', '10', '102190'), (3.5, 1))
(('1957', '04', '102190'), (7.3500000000000005, 1))
(('1957', '08', '102190'), (13.8, 1))
(('1957', '10', '102190'), (1.15, 1))
(('1958', '01', '102190'), (-12.6, 1))
(('1958', '04', '102190'), (-0.6000000000000001, 1))
```

```
(('1958', '05', '102190'), (8.9, 1))
(('1958', '07', '102190'), (16.9, 1))
```

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 andmaximumdaily precipitation is between 100 mm and 200mm.

In this exercise you will use the *temperature-readings.csv* and *precipitation-readings.csv* files.

The output should contain the following information:

Station number, maximum measured temperature, maximum daily precipitation

Code:

```
# import pyspark
from pyspark import SparkContext
import os
import sys

os.environ['PYSPARK_PYTHON'] = sys.executable
os.environ['PYSPARK_DRIVER_PYTHON'] = sys.executable
# create the spark application
sc = SparkContext(appName = "ex4")
# import the data
temperature = sc.textFile("data/temperature-readings.csv")
# transform the data by splitting each line
linesT = temperature.map(lambda line: line.split(";"))
# import the data
precipitation = sc.textFile("data/precipitation-readings.csv")
# transform the data by splitting each line
linesP= precipitation.map(lambda line: line.split(";"))

# map as (station_no, temp)
# find maximum read of station

station_temp= linesT. map(lambda x: (x[0],
float(x[3]))).reduceByKey(max).sortByKey()

station_temp2 = station_temp.filter(lambda x: x[1] > 20 and x[1] < 30 )
print(station_temp.take(20))
# calculate daily precipitation
# # map as (station, precipitation)
station_precip= linesP. map(lambda x:
(x[0],float(x[3]))).reduceByKey(max).sortByKey()
station_precip=station_precip.filter(lambda x: x[1]>100 and x[1]< 200)
```

```
station_precip_temp = station_temp.join(station_precip)
#
station_precip_temp.saveAsTextFile("./res/ex4")
```

Output:

No set satisfies the specified constraints.

```
{}
```

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 dothis, you will firstneed to calculate the total monthly precipitation for each station before calculating the monthly average (by averaging over stations).
In this exercise you will use the *precipitation-readings.csv* and *stations-Ostergotland.csv* files

The output should contain the following information:

Year,month, averagemonthly precipitation

Code:

```
from pyspark import SparkContext
import os
import sys

os.environ['PYSPARK_PYTHON'] = sys.executable
os.environ['PYSPARK_DRIVER_PYTHON'] = sys.executable


# create the spark application
sc = SparkContext(appName = "ex5")

precipitation_file = sc.textFile("data/precipitation-readings.csv")

# transform the data by splitting each line
lines_precipitation = precipitation_file.map(lambda line: line.split(";"))
```

```
# import stations in Ostergotland
Ostergotland = sc.textFile("data/stations-Ostergotland.csv")

# transform the data by splitting each line
lines_ost = Ostergotland.map(lambda line: line.split(";"))

# list od Ostergotland station
Ostergotland_ids = lines_ost.map(lambda x: x[0]).collect()

# map as ((station_no, yyyy, mm), (precipitation, 1))
# sum all values by reduceByKey
# map it again to find avg of month
# sort
ost_station_prec = lines_precipitation.map(lambda x: ((x[0], x[1][0:4],
x[1][5:7]), (float(x[3]), 1))). \
    filter(lambda x: x[0][0] in Ostergotland_ids and int(x[0][1])>1993 and
int(x[0][1])<2016).map(lambda x: ((x[0][1], x[0][2]), x[1])). \
    reduceByKey(lambda x,y: (x[0]+y[0], x[1]+y[1])). map(lambda x: (x[0],
x[1][0]/x[1][1])). sortByKey(False)

ost_station_prec.saveAsTextFile("./res/ex5")
```

Output:

```
[(('2015', '12'), 0.04152907394113418), (('2015', '11'), 0.09301182893539586),
(('2015', '10'), 0.00320183973111622), (('2015', '09'), 0.14667873303167459),
(('2015', '08'), 0.038253012048192765), (('2015', '07'), 0.1687267575703917),
(('2015', '06'), 0.1152353048892147), (('2015', '05'), 0.13079621185548954),
(('2015', '04'), 0.02228478023973846), (('2015', '03'), 0.05980701754385957),
(('2015', '02'), 0.03892591140729116), (('2015', '01'), 0.08375841303577748),
(('2014', '12'), 0.050399715757683324), (('2014', '11'), 0.0766447368421053),
(('2014', '10'), 0.10201520240410133), (('2014', '09'), 0.0705496905715325),
(('2014', '08'), 0.12998747539810362), (('2014', '07'), 0.03264690218356114),
(('2014', '06'), 0.10947004188672387), (('2014', '05'), 0.08136068735753119),
(('2014', '04'), 0.045907859078590685), (('2014', '03'), 0.05113636363636358),
(('2014', '02'), 0.06739256118712654), (('2014', '01'), 0.08681928546652816),
(('2013', '12'), 0.059056768558951936), (('2013', '11'), 0.0672954834028659),
(('2013', '10'), 0.07524441340782127), (('2013', '09'), 0.03793915248098513),
(('2013', '08'), 0.0770024919900321), (('2013', '07'), 0.07653866386112572),
(('2013', '06'), 0.08853997473380257), (('2013', '05'), 0.06701625589931826),
(('2013', '04'), 0.055479079876833856), (('2013', '03'), 0.010341207349081378),
(('2013', '02'), 0.03952003096574403), (('2013', '01'), 0.02993741307371343),
(('2012', '12'), 0.0906341683592866), (('2012', '11'), 0.09644111449309291),
(('2012', '10'), 0.08854635463546358), (('2012', '09'), 0.10160614525139666),
(('2012', '08'), 0.09249551971326166), (('2012', '07'), 0.07989179440937776),
(('2012', '06'), 0.18597889800703418), (('2012', '05'), 0.031375227686703075),
(('2012', '04'), 0.08789080727951468), (('2012', '03'), 0.01182572614107884),
(('2012', '02'), 0.04200244200244197), (('2012', '01'), 0.05939986360536479),
(('2011', '12'), 0.06046400382683561), (('2011', '11'), 0.0187645146307478),
```

```
(('2011', '10'), 0.061518631356925244), (('2011', '09'), 0.08143557965401492),
(('2011', '08'), 0.11670800450958284), (('2011', '07'), 0.1289336653837447),
```