Lab 1 - Big Data

Spark

johed883 and mikmo937

Notice that the "printed" outputs are fragments of the whole output.

Assignment 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.):

In [ ]:
```
#!/usr/bin/env python3
"""

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 and min
#max_temperatures = year_temperature.reduceByKey(lambda a,b: (a[0], max(a[1], b[1])))
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])

#min_temperatures = year_temperature.reduceByKey(lambda a,b: (a[0], min(a[1], b[1])))
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])

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

output/max:

('1975', 36.1) ('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)

output min:

('1990', -35.0) ('1952', -35.5) ('1974', -35.6) ('1954', -36.0) ('1992', -36.1) ('1975', -37.0) ('1972', -37.5) ('1995', -37.6) ('2000', -37.6) ('1957', -37.8) ('1983', -38.2)

Looks like the highest measured temperature between 1950-2014 was in 1975 and the highest of the coldest temperature of a year was in 1990.

Assignment 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 itappears 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:

In [ ]:
```
#!/usr/bin/env python3

from pyspark import SparkContext

sc = SparkContext(appName = "exercise 2")
# 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)
temperature = lines.map(lambda x: ((x[1][0:7]), (float(x[3]))))

#filter years
temperature = temperature.filter(lambda x: int(x[0][0:4])>=1950 and int(x[0][0:4])<=2014)

# filter temp
temperature = temperature.filter(lambda x: float(x[1])>10)

temperature = temperature.map(lambda x: (x[0],1)) # adding a 1 in the tuple to count

# counter
count_1 = temperature.reduceByKey(lambda a,b: a+b)

count_1.saveAsTextFile("BDA/output/ex2first")


# SECOND PART
# (key, value) = (year-month,station),(temperature)
temperature2 = lines.map(lambda x: ((x[1][0:7])+x[0]), (float(x[3]))))

#filter years
temperature2 = temperature2.filter(lambda x: int(x[0][0:4])>=1950 and int(x[0][0:4])<=2014)

# filter temp
temperature2 = temperature2.filter(lambda x: float(x[1])>10)

# mapping to add a 1 for our counter
temperature2 = temperature2.map(lambda x:(x[0],int(1)))

# reducing so we get unique keys with station
count_2 =  temperature2.reduceByKey(lambda a,b: a)

# mapping to remove stations
count_2  = count_2.map(lambda x:(x[0][0:7],x[1]))

# now counting the values
count_2 = count_2.reduceByKey(lambda a,b: a+b)

count_2.saveAsTextFile("BDA/output/ex2second")
```

output part 1 :

('1957-06', 18956) ('1959-04', 3866) ('1961-03', 1511) ('1962-06', 37819) ('1963-04', 2644) ('1965-06', 48744) ('1967-10', 17832) ('1969-09', 32722) ('1970-10', 9606) ('2000-08', 109201)

output part two:

('2000-08', 325) ('2001-10', 279) ('1961-03', 197) ('1970-10', 345) ('1989-09', 316) ('1996-06', 345) ('1990-03', 193) ('2003-05', 321) ('1959-04', 115) ('1992-04', 181) ('1990-09', 312)

assignment 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:

In [ ]:
```
#!/usr/bin/env python3

"""
ID; Year; Time; Temp; Quality

102170;2014-12-31;18:00:00;-8.7;G
"""

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(";"))

# create key of ID, Time with value temperature
month_temp = lines.map(lambda x: ((x[0], x[1][0:4], x[1][5:7], x[1][8:10]), float(x[3])))


# filter years to period 1960-2014
month_temp = month_temp.filter(lambda x: int(x[0][1]) >= 1960 and int(x[0][1]) <= 2014)


# get max and min temperature for each day and divide by (days in month)*2 for each station

# Get max and min
max_temperatures = month_temp.map(lambda x: (x[0],x[1])).reduceByKey(max)
min_temperatures = month_temp.map(lambda x: (x[0],x[1])).reduceByKey(min)

# Join max and min into same tuple and add them
join_temp = min_temperatures.join(max_temperatures)
join_temp = join_temp.map(lambda x: (x[0], x[1][0]+x[1][1]))

# Create new tuple with key as ID, Year, Month and value day, find maximum days for each key
max_date = join_temp.map(lambda x: ( (x[0][0], x[0][1], x[0][2]), x[0][3])).reduceByKey(max)

# Double the maximum day in accordance with average calculation
max_date = max_date.map(lambda x: (x[0], int(x[1])+int(x[1]) ))

# Reduce the original temp tuple to the same key of ID, Year, Month and have temperature as value, sum temperatures
join_temp = join_temp.map(lambda x : ( (x[0][0], x[0][1], x[0][2]), x[1])).reduceByKey(lambda x, y: x + y)

# Join the two tuples into one
avg_temp = max_date.join(join_temp)

# Divide summed temperatures with doubled maximum days per each month
avg_temp = avg_temp.map(lambda x: ((x[0]), x[1][1]/x[1][0]))

# Sorting
avg_temp = avg_temp.sortBy(ascending = False, keyfunc=lambda k: k[0])

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

Output:

(('99450', 2014', '12'), 1.9274193548387095)

(('99450', 2014', '11'), 5.931666666666667)

(('99450', 2014', '10'), 9.272580645161291)

(('99450', 2014', '09'), 13.77)

(('99450', 2014', '08'), 17.008064516129032)

(('99450', 2014', '07'), 18.51774193548387)

(('99450', 2014', '06'), 11.103333333333333)

(('99450', 2014', '05'), 7.633870967741936)

(('99450', 2014', '04'), 4.523333333333325)

(('99450', 2014', '03'), 2.9532258064516133)

(('99450', 2014', '02'), 1.7928571428571427)

(('99450', 2014', '01'), -1.05)

Assignment 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. In this exercise you will use the temperature-readings.csv and precipitation-readings.csv files. The output should contain the following information:

In [ ]:
```
#!/usr/bin/env python3

from pyspark import SparkContext

sc = SparkContext(appName = "exercise 2")
# This path is to the file on hdfs
temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
precipitation_file = sc.textFile("BDA/input/precipitation-readings.csv")

lines = temperature_file.map(lambda line: line.split(";"))
lines_pre = precipitation_file.map(lambda line: line.split(";"))

# (station) = (temperature)
temperature = lines.map(lambda x: ((x[0]),(float(x[3]))))
precipitation = lines_pre.map(lambda x:((x[0]),(float(x[3]))))

#filter out temps that arent between 25-30
temperature = temperature.filter(lambda x: x[1]>=25 and x[1]<=30)

precipitation = precipitation.filter(lambda x: x[1]>=100 and x[1]<=200)


#Get max
max_temperatures = temperature.reduceByKey(lambda a,b: a if a >= b else b)
max_pre = precipitation.reduceByKey(lambda a,b: a if a >= b else b)

# join the datasets on station
joined = max_pre.join(max_temperatures)


# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder
joined.saveAsTextFile("BDA/output/ex3")
```

The output from the previous code is empty as there are no matching stations in the two dataset after filtering on temperature between 25-30 and precipitation between 100-200mm.

Assignment 5

Calculate the average monthly precipitation for the Ostergotland region (list of stations is provided in the separate file) for the period 1993-2016. In order to do this, 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. HINT (not for the SparkSQL lab): Avoid using joins here! stations-Ostergotland.csv is small and if distributed will cause a number of unnecessary shuffles when joined with precipitationRDD. If you distribute precipitation-readings.csv then either repartition your stations RDD to 1 partition or make use of the collect function to acquire a python list and broadcast function to broadcast the list to all nodes. The output should contain the following information:

In [ ]:
```
#!/usr/bin/env python3
"""
precipitation-readings.csv:
(ID, Date, Time, Precipitation, Quality)
(99280, 2016-06-30; 21:00:00; 0.0; G)

stations-Ostergotland.csv:
(ID, Name, Measurement height, Lat, Long, Readings from, Readings to, Elevation)
(85270; Västerlösa; 2.0; 58.4447; 15.3772; 2002-05-01 00:00:00; 2011-02-28 23:59:59; 75.0)
"""

from pyspark import SparkContext

sc = SparkContext(appName = "exercise 1")

# This path is to the file on hdfs
# read station data and split
station_file = sc.textFile("BDA/input/stations-Ostergotland.csv")
s_lines = station_file.map(lambda line: line.split(";"))

 # select only station id and collect
ostergotland = s_lines.map(lambda x: int(x[0])).collect()

# reading precipitation data
precipitation_file = sc.textFile("BDA/input/precipitation-readings.csv")
p_lines = precipitation_file.map(lambda line: line.split(";"))

precip = p_lines.map(lambda x: (x[0], x[1], x[3])) # map ID, Time and precipitation as the others arent needed

# broadcast ostergotland so we can use it in RDD operations
bc = sc.broadcast(ostergotland)


precip = precip.filter(lambda x: int(x[1][0:4]) >= 1993 and int(x[1][0:4]) <= 2016) # filter to years 1993-2016
precip = precip.filter(lambda x: int(x[0]) in bc.value) # filter IDs to only those that are in Ostergötland


"""
(99280, 2016-06-30, 0.0)
"""

# aggregate precipitation by key for each month
precip = precip.map(lambda x: ((x[1],x[2]), int(x[0])))
total_prec = precip.map(lambda x:  (x[1], x[0][0][0:4], x[0][0][5:7]), float(x[0][1]) )).reduceByKey(lambda x,y: x + y)

# count average from total
avg_prec = total_prec.map(lambda x: ((x[0][1], x[0][2]), x[1])).groupByKey()
avg_prec = avg_prec.mapValues(lambda x: sum(x) / len(x))

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

Output:

(('1990', '12'), 39.550000000000003)

(('1997', '05'), 60.800000000000004)

(('1998', '07'), 85.166666666666664)

(('2003', '02'), 9.1166666666666665)

(('2004', '03'), 28.483333333333338)

(('2005', '09'), 13.950000000000001)

(('2005', '11'), 32.600000000000001)

(('2006', '01'), 17.683333333333334)

(('2007', '07'), 95.966666666666665)

(('2008', '01'), 44.966666666666667)