BDA1Eric

May 16, 2023

0.1 Assignments

0.1.1 Q1 What are thelowest 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,(station,temperature))
year_temperature = lines.map(lambda x: (x[1][0:4], (x[0],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[0], max(a[1],␣
 ↪b[1])))
max_temperatures = max_temperatures.sortBy(lambda x: x[1][1], ascending=False)

#Get max
min_temperatures = year_temperature.reduceByKey(lambda a,b: (a[0], min(a[1],␣
 ↪b[1])))
min_temperatures = min_temperatures.sortBy(lambda x: x[1][1], ascending=False)


max_temperatures_combine = max_temperatures.coalesce(1)
max_temperatures_combine = max_temperatures_combine.sortBy(lambda x: x[1][1],␣
 ↪ascending=False)
```

```
max_temperatures_combine.saveAsTextFile("BDA/output/l1max")

min_temperatures_combine = min_temperatures.coalesce(1)
min_temperatures_combine = min_temperatures_combine.sortBy(lambda x: x[1][1],␣
 ↪ascending=False)
min_temperatures_combine.saveAsTextFile("BDA/output/l1min")
```

- Output of l1max (u'1975', (u'102190', 36.1))
  (u'1992', (u'112080', 35.4))
  (u'1994', (u'123250', 34.7))
  (u'2014', (u'123340', 34.4))
  (u'2010', (u'123340', 34.4))
  (u'1989', (u'112080', 33.9))
  (u'1982', (u'133260', 33.8))
  (u'1968', (u'133470', 33.7))
  (u'1966', (u'102190', 33.5))
  (u'2002', (u'123250', 33.3))
  (u'1983', (u'123250', 33.3))
  (u'1986', (u'123250', 33.2))
  (u'1970', (u'112080', 33.2))
  (u'2000', (u'102190', 33.0))
  (u'1956', (u'108640', 33.0))
- Output of l1min (u'1990', (u'133260', -35.0))
  (u'1952', (u'108640', -35.5))
  (u'1974', (u'112080', -35.6))
  (u'1954', (u'134110', -36.0))
  (u'1992', (u'112080', -36.1))
  (u'1975', (u'123480', -37.0))
  (u'1972', (u'123480', -37.5))
  (u'1995', (u'102210', -37.6))
  (u'2000', (u'123250', -37.6))
  (u'1957', (u'123480', -37.8))
  (u'1983', (u'133260', -38.2))
  (u'1989', (u'112080', -38.2))
  (u'1953', (u'124020', -38.4))

0.1.2 Q2 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.

[ ]:
```
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), temp)
year_month_temperature = lines.map(lambda x: ((x[1][0:4],x[1][5:
 ↪7],x[0]),float(x[3])))


#filter
year_month_temperature = year_month_temperature.filter(lambda x: int(x[0][0])␣
 ↪>= 1950 and int(x[0][0]) <=2014 and x[1] > 10)
#count = year_month_temperature.map(lambda x: (x[0], 1))
count = year_month_temperature.map(lambda x: ((x[0][0],x[0][1]), 1))
count = count.reduceByKey(lambda a, b: a + b)
count = count.coalesce(1)
count_sort = count.sortByKey().sortByKey(1)
count_sort.saveAsTextFile("BDA/output/countsort")
########################



count_distinct = year_month_temperature.map(lambda x: (x[0],1)).distinct()
count_distinct = count_distinct.map(lambda x: ((x[0][0],x[0][1]), 1))
count_distinct = count_distinct.reduceByKey(lambda a, b: a + b)
count_distinct = count_distinct.coalesce(1)
count_distinct_sort = count_distinct.sortByKey().sortByKey(1)
count_distinct_sort.saveAsTextFile("BDA/output/count_distinct_sort")
```

- Output of countsort (the number of readings for each month) ((u'1950', u'03'), 81)
  ((u'1950', u'04'), 352)
  ((u'1950', u'05'), 2802)
  ((u'1950', u'06'), 4886)
  ((u'1950', u'07'), 5811)
  ((u'1950', u'08'), 5954)
  ((u'1950', u'09'), 3612)
  ((u'1950', u'10'), 1248)
  ((u'1950', u'11'), 2)
  ((u'1950', u'12'), 1)
  ((u'1951', u'02'), 1)
- Output of count_distinct_sort(the number of distinct readings for each month) ((u'1950', u'03'), 26)
  ((u'1950', u'04'), 36)
  ((u'1950', u'05'), 46)
  ((u'1950', u'06'), 47)
  ((u'1950', u'07'), 49)
  ((u'1950', u'08'), 49)
  ((u'1950', u'09'), 50)

((u'1950', u'10'), 46)
((u'1950', u'11'), 2)
((u'1950', u'12'), 1)

0.1.3 Q4 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.

[ ]:
```
from pyspark import SparkContext

sc = SparkContext(appName = "exercise 4")
# 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")

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

get_temperature = temperature_lines.map(lambda x: (x[0],float(x[3])))
get_percipitation = precipitation_file.map(lambda x: (x[0],float(x[3])))

max_temp = get_temperature.reduceByKey(max)
filter_temp = max_temp.filter(lambda x : x[1]>25 and x[1]<30)


max_perc = get_percipitation.reduceByKey(max)
filter_perc = max_perc.filter(lambda x : x[1]>1000 and x[1]<200)



join_output= filter_temp.join(filter_perc)
join_output = join_output.coalesce(1)
join_output_sort = join_output.sortByKey()
join_output_sort.saveAsTextFile("BDA/output/temp_perce_sort")

### Below are for testing since no output
#filter_perc = max_perc.filter(lambda x : x[1]>0 and x[1]<15)
#filter_perc = max_perc.filter(lambda x : x[1]>100)
#filter_temp.saveAsTextFile("BDA/output/temp_test")
#max_perc.saveAsTextFile("BDA/output/maxperc_test")
#filter_perc.saveAsTextFile("BDA/output/filterperc_test")
```

- The output for temp_perce_sort is empty, since no data meet the criteria