BDA1_report_ver2

May 23, 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,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 or␣
 ↪int(x[0])<=2014)

#Get max
max_temperatures = year_temperature.reduceByKey(lambda a, b: (a[0], a[1]) if␣
 ↪a[1] > b[1] else (b[0],b[1]))
#max_temperatures = max_temperatures.sortBy(ascending = False, keyfunc=lambda k:
 ↪ k[1])


#get min
min_temperatures = year_temperature.reduceByKey(lambda a, b: (a[0], a[1]) if␣
 ↪a[1] < b[1] else (b[0],b[1]))
#min_temperatures = min_temperatures.sortBy(ascending = False, keyfunc=lambda k:
 ↪ k[1])

joineddf = max_temperatures.join(min_temperatures)
```

```
joineddf = joineddf.sortBy(ascending = False, keyfunc = lambda k : k[1][0][1])

#print(max_temperatures.collect())

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

- Output of Q1 :

Year Station, max_temp Station,min_temp (u'1975', ((u'86200', 36.1), (u'157860', -37.0)))
(u'1992', ((u'63600', 35.4), (u'179960', -36.1)))
(u'1994', ((u'117160', 34.7), (u'179960', -40.5)))
(u'2014', ((u'96560', 34.4), (u'192840', -42.5)))
(u'2010', ((u'75250', 34.4), (u'191910', -41.7)))
(u'1947', ((u'53770', 34.3), (u'139570', -32.0)))
(u'1989', ((u'63050', 33.9), (u'166870', -38.2)))
(u'1982', ((u'94050', 33.8), (u'113410', -42.2)))
(u'1968', ((u'137100', 33.7), (u'179950', -42.0)))
(u'1966', ((u'151640', 33.5), (u'179950', -49.4)))
(u'1945', ((u'85600', 33.4), (u'139570', -26.3)))
(u'2002', ((u'78290', 33.3), (u'169860', -42.2)))
(u'1983', ((u'98210', 33.3), (u'191900', -38.2)))
(u'1986', ((u'76470', 33.2), (u'167860', -44.2)))
(u'1970', ((u'103080', 33.2), (u'179950', -39.6)))
(u'2015', ((u'125170', 33.1), (u'179960', -39.9)))
(u'1956', ((u'145340', 33.0), (u'160790', -45.0)))
(u'2000', ((u'62400', 33.0), (u'169860', -37.6)))
(u'1959', ((u'65160', 32.8), (u'159970', -43.6)))
(u'1991', ((u'137040', 32.7), (u'179960', -39.3)))
(u'2006', ((u'75240', 32.7), (u'169860', -40.6)))
(u'1988', ((u'102540', 32.6), (u'170790', -39.9)))
(u'2011', ((u'172770', 32.5), (u'179960', -42.0)))
(u'1999', ((u'98210', 32.4), (u'192830', -49.0)))
(u'1948', ((u'53770', 32.4), (u'139570', -30.0)))
(u'1955', ((u'97260', 32.2), (u'160790', -41.2)))
(u'2007', ((u'86420', 32.2), (u'169860', -40.7)))
(u'2008', ((u'102390', 32.2), (u'179960', -39.3)))
(u'2003', ((u'136420', 32.2), (u'179960', -41.5)))

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)

Year, month, count (sorted by date) ((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)

Year, month, count (sorted by date) ((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 Q3.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

[ ]:
```
from pyspark import SparkContext

sc = SparkContext(appName = 'exercise 3')
temperature_file = sc.textFile('BDA/input/temperature-readings.csv')

lines = temperature_file.map(lambda line: line.split(';'))

year_month_date_station_temp = lines.map(lambda x: ( (x[1][0:4],x[1][5:
 ↪7],x[1][8:],x[0]) , (float(x[3])) ) )

year_month_date_station_temp =year_month_date_station_temp.filter(lambda x :␣
 ↪int(x[0][0])>=1960 and int(x[0][0])<=2014)

min_max_temperatures = year_month_date_station_temp.groupByKey()
min_max_temperatures = min_max_temperatures.mapValues(lambda x: (min(x),max(x)))

# calculating daily average
avg_temperature = min_max_temperatures.map(lambda x: ((x[0][0],␣
 ↪x[0][1],x[0][3]), (x[1][0] + x[1][1]) / 2))

#add count column
avg_temperature = avg_temperature.mapValues(lambda x : (x,1))

avg_monthly_temperature = avg_temperature.reduceByKey(lambda a,b : (a[0] +␣
 ↪b[0],a[1] + b[1]) )
avg_monthly_temperature = avg_monthly_temperature.mapValues(lambda x : (x[0]/
 ↪x[1]))
```

```
avg_monthly_temperature.saveAsTextFile('BDA/output/')
```

- Output of Q3

Year, month, station number, average monthly temperature ((u'1989', u'06', u'92400'), 14.686666666666666)
((u'1982', u'09', u'107530'), 11.171666666666669)
((u'2002', u'11', u'136360'), -5.861666666666666)
((u'1967', u'08', u'98170'), 15.408064516129032)
((u'2002', u'08', u'181900'), 15.598387096774195)
((u'1981', u'11', u'63440'), 3.086666666666667)
((u'1996', u'08', u'96190'), 17.1)
((u'1994', u'06', u'71180'), 13.036666666666669)
((u'2010', u'10', u'64130'), 5.974193548387096)
((u'1995', u'06', u'62400'), 16.00166666666667)
((u'1972', u'10', u'64130'), 7.666129032258065)
((u'1985', u'02', u'81130'), -7.678571428571428)
((u'1977', u'10', u'191900'), -2.9322580645161294)
((u'1988', u'04', u'86330'), 4.626666666666666)
((u'1989', u'04', u'180940'), -0.43333333333333346)
((u'1980', u'02', u'123250'), -14.946551724137931)
((u'1964', u'04', u'53640'), 7.694999999999999)
((u'1984', u'05', u'106100'), 10.933870967741937)
((u'2002', u'09', u'178860'), 6.408333333333333)
((u'1977', u'08', u'182930'), 10.193548387096774)
((u'1983', u'02', u'78240'), -2.2660714285714287)
((u'1967', u'10', u'162880'), 3.229032258064516)
((u'1990', u'02', u'89240'), 3.692857142857143)
((u'1966', u'04', u'137110'), 0.23500000000000004)
((u'1990', u'07', u'52360'), 16.559677419354838)
((u'2000', u'05', u'73470'), 12.243548387096775)
((u'1979', u'01', u'123480'), -19.470967741935485)
((u'1985', u'08', u'95160'), 15.506451612903227)
((u'1970', u'01', u'83340'), -7.161290322580644)

0.1.4 Q4 Provide a list of stations with their associated maximum measured temperatures and precipitation

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])))
#(station, date, precipitation)
get_precipitation = precipitation_file.map(lambda x: ((x[0], x[1]),␣
 ↪float(x[3])))
sumdaily_precipitation = get_precipitation.reduceByKey(lambda x, y: x + y)
#map it with (station, precipitation)
sumdaily_precipitation = sumdaily_precipitation.map(lambda x: ((x[0][0]),␣
 ↪float(x[1])))

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


max_prec = sumdaily_precipitation.reduceByKey(max)

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


join_output= filter_temp.join(filter_prec)
join_output = join_output.coalesce(1)
join_output_sort = join_output.sortByKey()
join_output_sort.saveAsTextFile("BDA/output/temp_prec_sort")
```

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

0.1.5 Q5. Calculate the average monthly precipitation for the Östergotland region

(list of stations is provided in the separate file) for the period 1993-2016. In orderto dothis, you will first need 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:

Year, month, average monthly precipitation

[ ]:
```
from pyspark import SparkContext

sc = SparkContext(appName = 'exercise 3')
precipitaion_file = sc.textFile('BDA/input/precipitation-readings.csv')
stations_file = sc.textFile('BDA/input/stations-Ostergotland.csv')
```

```
lines = precipitaion_file.map(lambda line: line.split(';'))
stations = stations_file.map(lambda line: line.split(';'))

# extracting only the station numbers, then collecting and broadcasting to make␣
 ↪available to all nodes to filter later
stations = stations.map(lambda x: x[0])
stations = stations.collect()
stations = sc.broadcast(stations).value

year_month_station_precip = lines.map(lambda x: ( (x[1][0:4],x[1][5:7],x[0]) ,␣
 ↪(float(x[3])) ) )
#filter for years
year_month_station_precip =year_month_station_precip.filter(lambda x :␣
 ↪int(x[0][0])>=1993 and int(x[0][0])<=2016)

#filter for stations in Ostergotland
year_month_station_precip = year_month_station_precip.filter(lambda x : x[0][2]␣
 ↪in stations)

#summing up to get total precipitation per month,station and year
year_month_station_precip = year_month_station_precip.reduceByKey(lambda a,b:␣
 ↪a+b)

# remap to add count column in value
monthly_precipitation = year_month_station_precip.map(lambda x:␣
 ↪((x[0][0],x[0][1]),(x[1],1)))

#summing up
monthly_precipitation = monthly_precipitation.reduceByKey(lambda a,b: (a[0] +␣
 ↪b[0],a[1] + b[1]))

#obtaining average
avg_monthly_precipitaion= monthly_precipitation.mapValues(lambda x : (x[0]/
 ↪x[1]))


avg_monthly_precipitaion.saveAsTextFile('BDA/output/')
```

- Output of Q5 ### Year, month, average monthly precipitation

((u'2012', u'09'), 72.75)
((u'1995', u'05'), 26.00000000000002)
((u'1996', u'12'), 39.55000000000003)
((u'2011', u'08'), 86.26666666666667)
((u'2007', u'04'), 21.249999999999996)
((u'2007', u'06'), 108.94999999999999)
((u'1993', u'04'), 0.0)

((u'2011', u'10'), 43.75)
((u'2014', u'10'), 72.13749999999999)
((u'1996', u'09'), 57.46666666666667)
((u'1995', u'07'), 43.6)
((u'2002', u'05'), 72.13333333333334)
((u'2010', u'04'), 23.78333333333333)
((u'1999', u'01'), 61.933333333333394)
((u'2013', u'11'), 46.37500000000002)
((u'2016', u'05'), 29.250000000000004)
((u'1999', u'10'), 18.549999999999997)

[ ]: