BDA Lab-1

Aswath Mandakath Gopinath(aswma317), Singapura Ravi Varun(varsi146)

Question 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

Answer:

Code:

```
from pyspark import SparkContext

sc = SparkContext(appName = "BDA1-1")

# 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], a[1]) if(a[1] > b[1]) else (b[0], b[1]))

#Get min

min_temperatures = year_temperature.reduceByKey(lambda a,b: (b[0], b[1]) if(a[1] > b[1]) else (a[0], a[1]))

#Get union of max and min

final_temperatures = sc.union([max_temperatures, min_temperatures])

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

#print(max_temperatures.collect())

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder

final_temperatures.saveAsTextFile("BDA/output")
```

Output:

```
================= FINAL OUTPUT
(u'1975', (u'86200', 36.1))
(u'1992', (u'63600', 35.4))
(u'1994', (u'117160', 34.7))
(u'2014', (u'96560', 34.4))
(u'2010', (u'75250', 34.4))
(u'1989', (u'63050', 33.9))
(u'1982', (u'94050', 33.8))
(u'1968', (u'137100', 33.7))
(u'1966', (u'151640', 33.5))
(u'1983', (u'98210', 33.3))
(u'2002', (u'78290', 33.3))
(u'1970', (u'103080', 33.2))
```

```
(u'1980', (u'191900', -45.0))
(u'1956', (u'160790', -45.0))
(u'1967', (u'166870', -45.4))
(u'1987', (u'123480', -47.3))
(u'1978', (u'155940', -47.7))
(u'1999', (u'192830', -49.0))
(u'1966', (u'179950', -49.4))
```

Question 2.

Count the number of readings for each month in the period of 1950-2014 which are higher than10 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:

Year, month, count

Answer:

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

#filter
```

```
year_mon_temperature = year_mon_temperature.filter(lambda x: int(x[0][0])>=1950 and int(x[0][0])<=2014 and int(x[1]) >= 10)

#Get count- 1st question ((year, month), count)

count_temp = year_mon_temperature.map(lambda a: ((a[0][0],a[0][1]),1)).reduceByKey(lambda a,b: a+b) # this line is commented out while getting distinct counts.

#Get count- 2nd question ((year, month, station), count)

count_temp = year_mon_temperature.reduceByKey(lambda a,b: 1)

#Change the tuple - ((year, month), 1) and get the count

count_temp = count_temp.map(lambda a: ((a[0][0],a[0][1]),1)).reduceByKey(lambda a,b: a+b)

#print(final_temp.collect())

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder

count_temp.saveAsTextFile("BDA/output")
```

Output:

Part-1 Output: Getting counts of readings of each month greater than 10 degrees.

```
================= FINAL OUTPUT
((u'2008', u'11'), 1663)
((u'1970', u'01'), 1)
((u'1971', u'06'), 47448)
((u'1989', u'12'), 32)
((u'1966', u'11'), 213)
((u'1956', u'05'), 12696)
((u'1998', u'07'), 120912)
((u'1975', u'02'), 33)
((u'1983', u'10'), 12074)
((u'1992', u'05'), 34325)
```

Part-2 Output: Getting counts of distinct readings of each station & month greater than 10 degrees.

```
================= FINAL OUTPUT
((u'2008', u'11'), 107)
((u'1970', u'01'), 1)
((u'1992', u'05'), 311)
((u'1971', u'06'), 374)
((u'1989', u'12'), 10)
((u'1966', u'11'), 76)
((u'1956', u'05'), 125)
((u'1998', u'07'), 326)
((u'1975', u'02'), 19)
((u'1983', u'10'), 246)
((u'1978', u'09'), 358)
((u'1969', u'10'), 346)
((u'1994', u'10'), 265)
```

Question 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

Answer:

Code:

```
import calendar

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) = ((day, station), temperature)

day_temp = lines.map(lambda x: ((x[1][0:10],x[0]) ,float(x[3])))

max_temp = day_temp.reduceByKey(lambda a,b: max(a, b))

min_temp = day_temp.reduceByKey(lambda a,b: min(a, b))

# get the union of rdds

final_temp = sc.union([max_temp, min_temp])

#change ((day, station), temperature) to ((year, month, station), temp) and filter

year_mon_temperature = final_temp.map(lambda a: ((a[0][0][0:4], a[0][0][5:7], a[0][1]), a[1]))

#filter step

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

#Sum up the max and mins for each ((year, month, station), temp)

year_mon_temperature = year_mon_temperature.reduceByKey(lambda a,b: a+b)

#Divide by the number of days in the month

year_mon_temperature = year_mon_temperature.map(lambda a: ((a[0][0], a[0][1], a[0][2]),
round(a[1]/(calendar.monthrange(int(a[0][0]), int(a[0][1]))[1]*2), 2)))

#year_mon_temperature = year_mon_temperature.map(lambda a: ((a[0][0], a[0][1], a[0][2]),
round(a[1]/62, 2)))

#Sort based on the key

year_mon_temperature = year_mon_temperature.sortBy(ascending = False, keyfunc=lambda k: k[0])


# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder

year_mon_temperature.saveAsTextFile("BDA/output")
```

Output:

```
================= FINAL OUTPUT =====
((u'2014', u'12', u'99450'), 1.93)
((u'2014', u'12', u'99280'), 2.31)
((u'2014', u'12', u'99270'), 2.36)
((u'2014', u'12', u'98490'), -1.62)
((u'2014', u'12', u'98290'), 0.52)
((u'2014', u'12', u'98230'), 0.33)
((u'2014', u'12', u'98210'), 0.57)
((u'2014', u'12', u'98180'), 0.03)
((u'2014', u'12', u'98040'), 0.39)
((u'2014', u'12', u'97530'), -1.79)
((u'2014', u'12', u'97510'), -1.22)
((u'2014', u'12', u'97400'), -0.9)
((u'2014', u'12', u'97370'), -1.57)
```

```
((u'2013', u'12', u'192840'), -10.54)
((u'2013', u'12', u'191910'), -11.12)
((u'2013', u'12', u'191720'), -10.61)
((u'2013', u'12', u'189720'), -9.26)
```

Question 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:

Station number, maximum measured temperature, maximum daily precipitation

Answer:

Code:

```
import calendar

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")

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

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

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

# (key, value) = (station, temp) and ((station,year,month,day), precip))

max_temp_station = temp_lines.map(lambda x: (x[0], float(x[3])))
```

```
max_precip_station =  precip_lines.map(lambda x: ((x[0],x[1][:4], x[1][5:7],x[1][8:10]), float(x[3])))

# ((station,year,month,day), precip))

max_precip_station = max_precip_station.reduceByKey(lambda a,b: a+b)

# (station, precip) but duplicates

max_precip_station = max_precip_station.map(lambda x: (x[0][0], x[1]))

#get max_temp for (station, maxtemp) and (station, maxprecip)

max_temp_station = max_temp_station.reduceByKey(lambda a,b: max(a,b))

max_precip_station = max_precip_station.reduceByKey(lambda a,b: max(a,b))

# filter the (station, temp) and (station, precip)

max_temp_station = max_temp_station.filter(lambda x: float(x[1])>=25.0 and float(x[1])<=30.0)

max_precip_station = max_precip_station.filter(lambda x: float(x[1])>=100.0 and float(x[1])<=200.0)

#final result by joining by station

result = max_temp_station.join(max_precip_station)

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder

result.saveAsTextFile("BDA/output")
```

Output:

There is no final output, since, for the maximum temperature readings in between 25 and 30 degrees and maximum daily precipitation in between 100 mm and 200mm, there are no stations listed.

Question 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 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. The output should contain the following information:

Year, month, average monthly precipitation

Answer:

Code: Updated the code as per comments by now using broadcast function.

```
import calendar

from pyspark import SparkContext

sc = SparkContext(appName = "exercise 1")

# This path is to the file on hdfs

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

stations_file = sc.textFile("BDA/input/stations-Ostergotland.csv")
```

```
station_lines = stations_file.map(lambda line: line.split(";"))

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

#Get stations and broadcast

stations = sc.broadcast(station_lines.map(lambda x: x[0]).collect())

#(key, value) = ((station, year, month), precip)

mon_precip = precip_lines.map(lambda x: ((x[0],int(x[1][:4]), int(x[1][5:7])),float(x[3]))) \

    .filter(lambda a: a[0][0] in stations.value)

#Sum up the precipitation ((station, year, month), sum_month_precip)

year_mon_precip = mon_precip.reduceByKey(lambda a,b: a+b)

#filter step

year_mon_precip = year_mon_precip.filter(lambda x: x[0][1]>=1993 and x[0][1]<=2016)

#filter by station and transform ((station, year, month), (precip, count))

year_mon_precip = year_mon_precip.map(lambda a: ((a[0][0], a[0][1], a[0][2]), (a[1], 1)))

#Removing station - transforming to ((year, month), (avg_precip, count_stn)) Updated Code:

year_mon_precip = year_mon_precip.map(lambda x: ((x[0][1], x[0][2]),x[1])).reduceByKey(lambda
a,b: (a[0]+b[0],a[1]+b[1])).map(lambda x: ((x[0][0], x[0][1]),round(x[1][0]/x[1][1],3)))

#sort

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

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder

year_mon_precip.saveAsTextFile("BDA/output")
```

Output:

```
================ FINAL OUTPUT
((2006, 8), 148.083)
((2008, 8), 138.517)
((2000, 7), 135.867)
((1995, 9), 134.55)
((2012, 6), 132.2)
((2015, 7), 119.1)
((2006, 10), 118.167)
((2003, 7), 113.467)
((2009, 7), 113.167)
((2001, 9), 110.633)
((2000, 10), 110.3)
((2007, 6), 108.95)
((2000, 11), 108.117)
((2010, 8), 108.05)
((2005, 7), 104.35)
((2015, 9), 101.3)
((2002, 6), 98.783)
((1995, 6), 97.2)
```