BigDataAnalysis BDA1

Helena Llorens Lluís (hllor282) Yi Yang (yiyan338)

April 2025

1 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.):
Year, temperature
Please notice that filtering before the reduce step will save the time and resource for running your program.

- code

```
from pyspark import SparkContext

#step 1 : create a SparkContext which tells Spark how to access a cluster
sc = SparkContext(appName = "exercise 1")

#step 2 : create distributed datasets
# 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(";"))

#step 3 : RDD operation
# (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 and
↪  int(x[0])<=2014)

#Get max
#max_temperatures = year_temperature.reduceByKey(lambda a,b: (a[0], max(a[1],
↪  b[1])))
#max_temperatures =year_temperature.reduceByKey(max)
max_temperatures = year_temperature.reduceByKey(lambda a,b: a if a[1]>= b[1] else
↪  b )
```

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

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


print(" Maxmimum : ")
print(max_temperatures.collect())

print("Minimum : ")
print(min_temperatures.collect())

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

- Maximum

```
[
    ('1975', ('86200', 36.1)), ('1992', ('63600', 35.4)),
    ('1994', ('117160', 34.7)), ('2014', ('96560', 34.4)),
    ('2010', ('75250', 34.4)), ('1989', ('63050', 33.9)),
    ('1982', ('94050', 33.8)), ('1968', ('137100', 33.7)),
    ('1966', ('151640', 33.5)), ('2002', ('78290', 33.3)),
    ('1983', ('98210', 33.3)), ('1986', ('76470', 33.2)),
    ('1970', ('103080', 33.2)), ('2000', ('62400', 33.0)),
    ('1956', ('145340', 33.0)), ('1959', ('65160', 32.8)),
    ('2006', ('75240', 32.7)), ('1991', ('137040', 32.7)),
    ('1988', ('102540', 32.6)), ('2011', ('172770', 32.5)),
    ('1999', ('98210', 32.4)), ('2008', ('102390', 32.2)),
    ('1955', ('97260', 32.2)), ('2003', ('136420', 32.2)),
    ('2007', ('86420', 32.2)), ('1953', ('65160', 32.2)),
    ('1973', ('71470', 32.2)), ('2005', ('107140', 32.1)),
    ('1979', ('63600', 32.0)), ('1969', ('71470', 32.0)),
    ('2001', ('62400', 31.9)), ('1997', ('74180', 31.8)),
    ('1977', ('94180', 31.8)), ('2013', ('98210', 31.6)),
    ('2009', ('81370', 31.5)), ('2012', ('54990', 31.3)),
    ('1964', ('76430', 31.2)), ('1972', ('137100', 31.2)),
    ('1971', ('65130', 31.2)), ('1976', ('75040', 31.1)),
    ('1961', ('76000', 31.0)), ('1963', ('62410', 31.0)),
    ('1958', ('117160', 30.8)), ('1996', ('96140', 30.8)),
    ('1995', ('76420', 30.8)), ('1978', ('71470', 30.8)),
    ('1974', ('167710', 30.6)), ('1954', ('53650', 30.5)),
    ('1952', ('106100', 30.4)), ('1980', ('161790', 30.4)),
    ('1990', ('54330', 30.2)), ('2004', ('117160', 30.2)),
    ('1985', ('85450', 29.8)), ('1957', ('75040', 29.8)),
    ('1993', ('132030', 29.7)), ('1981', ('65160', 29.7)),
    ('1987', ('98210', 29.6)), ('1984', ('105370', 29.5)),
    ('1967', ('75040', 29.5)), ('1960', ('173810', 29.4)),
    ('1950', ('75040', 29.4)), ('1998', ('63600', 29.2)),
    ('1965', ('116500', 28.5)), ('1951', ('75040', 28.5)),
    ('1962', ('86200', 27.4))
]
```

- Minimum

```
[
    ('1990', ('147270', -35.0)), ('1952', ('192830', -35.5)),
    ('1974', ('179950', -35.6)), ('1954', ('113410', -36.0)),
    ('1992', ('179960', -36.1)), ('1975', ('157860', -37.0)),
    ('1972', ('167860', -37.5)), ('1995', ('182910', -37.6)),
    ('2000', ('169860', -37.6)), ('1957', ('159970', -37.8)),
    ('1989', ('166870', -38.2)), ('1983', ('191900', -38.2)),
    ('1953', ('183760', -38.4)), ('2009', ('179960', -38.5)),
    ('1993', ('191900', -39.0)), ('1984', ('191900', -39.2)),
    ('2008', ('179960', -39.3)), ('1973', ('166870', -39.3)),
    ('1991', ('179960', -39.3)), ('2005', ('155790', -39.4)),
    ('1964', ('166810', -39.5)), ('1961', ('181900', -39.5)),
    ('1970', ('179950', -39.6)), ('2004', ('166940', -39.7)),
    ('1988', ('170790', -39.9)), ('1960', ('167710', -40.0)),
    ('1997', ('179960', -40.2)), ('1994', ('179960', -40.5)),
    ('2006', ('169860', -40.6)), ('2013', ('179960', -40.7)),
    ('2007', ('169860', -40.7)), ('1963', ('181900', -41.0)),
    ('1955', ('160790', -41.2)), ('2003', ('179960', -41.5)),
    ('1969', ('181900', -41.5)), ('1996', ('155790', -41.7)),
    ('2010', ('191910', -41.7)), ('1950', ('155910', -42.0)),
    ('1962', ('181900', -42.0)), ('2011', ('179960', -42.0)),
    ('1951', ('155910', -42.0)), ('1968', ('179950', -42.0)),
    ('1976', ('192830', -42.2)), ('1982', ('113410', -42.2)),
    ('2002', ('169860', -42.2)), ('1977', ('179950', -42.5)),
    ('2014', ('192840', -42.5)), ('1998', ('169860', -42.7)),
    ('2012', ('191910', -42.7)), ('1958', ('159970', -43.0)),
    ('1985', ('159970', -43.4)), ('1959', ('159970', -43.6)),
    ('1981', ('166870', -44.0)), ('1979', ('112170', -44.0)),
    ('2001', ('112530', -44.0)), ('1965', ('189780', -44.0)),
    ('1986', ('167860', -44.2)), ('1971', ('166870', -44.3)),
    ('1980', ('191900', -45.0)), ('1956', ('160790', -45.0)),
    ('1967', ('166870', -45.4)), ('1987', ('123480', -47.3)),
    ('1978', ('155940', -47.7)), ('1999', ('192830', -49.0)),
    ('1966', ('179950', -49.4))
]
```

2 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 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

```
#!/usr/bin/env python3

from pyspark import SparkContext

#step 1: create a SparkContext which tells Spark how to access a cluster
sc = SparkContext(appName = "exercise 1")

# Step 2: Create distributed datasets
# 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(";"))

#step 3: RDD operation
# (key, value) = (year,temperature)
#year_temperature = lines.map(lambda x: (x[1][0:4], (x[0],float(x[3]))))

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

#filter: year[1950,2014],temp>10
yearMonth_temperature = yearMonth_temperature.filter(lambda x:
↪  int(x[0][0:4])>=1950 and int(x[0][0:4])<=2014)
yearMonth_temperature = yearMonth_temperature.filter(lambda x : x[1][1]> 10)

#1.
#(year-month,1)
yearMonth = yearMonth_temperature.map(lambda x : (x[0],1))
#count per year
count=yearMonth.reduceByKey(lambda a,b: a+b)


#2.
#count distinct readings  from each station per(year, month)
yearMonth_distinct = yearMonth_temperature.map(lambda x: (x[0], x[1][0]))
#using distinct by stationn_id for each month
yearMonth_distinct =yearMonth_distinct.distinct()
#(year-month,1)
yearMonth_distinct=yearMonth_distinct.map(lambda x : (x[0],1))
#count per year
count_distinct=yearMonth_distinct.reduceByKey(lambda a,b: a+b)



print("Count of readings(temperature higher than 10 degree) per month: ")
print(count.collect())

print("Count of distinct readings(temperature higher than 10 degree) per month:")
print(count_distinct.collect())

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder
count.saveAsTextFile("BDA/output/task1")
count_distinct.saveAsTextFile("BDA/output/task2")
```

- Count of readings(temperature higher than 10 degrees) per month *(partial results)*

```
[
    ('2000-08', 109201), ('2001-10', 43671), ('2003-05', 48264),
    ('2008-10', 26107), ('2010-03', 506), ('2010-08', 124417),
    ('2013-09', 81960), ('1951-09', 9601), ('1953-03', 427),
    ('1953-04', 1871), ('1957-06', 18956), ('1959-04', 3866),
    ('1961-03', 1511), ('1962-06', 37819), ('1963-04', 2644),
    ('1965-06', 48744), ('1969-09', 32722), ('1970-10', 9606),
    ('1982-04', 4172), ('1983-07', 56777), ('1988-06', 63572),
    ('1989-09', 50222), ('1990-03', 3455), ('1990-09', 34171),
    ('1994-05', 21529), ('1995-09', 46040), ('1967-10', 17832),
    ('1996-06', 80440), ('1990-02', 1160), ('1992-04', 1688),
    ('1963-03', 1), ('1983-11', 596), ('1995-03', 102),
    ('1974-11', 33), ('1954-11', 25), ('1983-03', 23),
    ('1995-11', 60), ('1950-12', 1), ('1960-01', 1),
    ('1962-03', 1), ('1958-01', 1), ('1991-01', 2),
    ('1952-11', 1), ('1981-05', 35371)
]
```

- Count of distinct readings(temperature higher than 10 degree) per month: *(partial results)*

```
[
     ('1996-06', 345), ('2000-08', 325), ('2001-10', 279),
    ('2003-05', 321), ('2008-10', 226), ('2010-08', 318),
    ('2013-09', 299), ('1961-03', 197), ('1962-06', 297),
    ('1963-04', 283), ('1965-06', 355), ('1967-10', 324),
    ('1969-09', 359), ('1970-10', 345), ('1982-04', 246),
    ('1983-07', 319), ('1988-06', 322), ('1989-09', 316),
    ('1990-02', 148), ('1990-03', 193), ('1990-09', 312),
    ('1992-04', 181), ('1994-05', 299), ('1995-09', 315),
    ('2010-03', 65), ('1983-11', 160), ('1959-04', 115),
    ('1995-03', 59), ('1951-09', 112), ('1953-03', 77),
    ('1953-04', 104), ('1957-06', 128), ('1995-11', 24),
    ('1954-11', 21), ('1983-03', 17), ('1974-11', 19),
    ('1991-01', 1), ('1958-01', 1), ('1952-11', 1),
    ('1963-03', 1), ('1962-03', 1), ('1960-01', 1),
    ('1950-12', 1), ('1997-10', 229), ('1998-03', 142),
    ('1998-09', 326), ('2002-08', 322), ('2004-09', 321),
    ('2005-07', 307), ('2005-08', 306), ('2010-05', 319),
    ('2012-09', 306), ('1950-04', 36), ('1951-05', 98),
    ('1952-04', 107), ('1956-05', 125), ('1956-10', 103),
    ('1958-10', 124), ('1959-06', 128), ('1960-07', 126),
    ('1967-06', 359), ('1967-07', 351), ('1968-04', 322),
    ('1975-05', 367), ('1975-08', 367), ('1981-05', 337)
]
```

3 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:
Year, month, station number, average monthly temperature

- code

```
#!/usr/bin/env python3

from pyspark import SparkContext

#step 1 : create a SparkContext which tells Spark how to access a cluster
sc = SparkContext(appName = "exercise 1")

#step 2 : create distributed datasets
# 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(";"))

#step 3 : RDD operation
# (key, value) = (station-year-month-day,temperature)
#((102170,2013-11-01),6.3)

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

#filter: year[1960,2014]
temperature = temperature.filter(lambda x: int(x[0][1][0:4])>=1960 and
↪  int(x[0][1][0:4])<=2014)

#get daily maximum and minimum
daily_max = temperature.reduceByKey(lambda a,b: max(a,b))
daily_min = temperature.reduceByKey(lambda a,b : min(a,b))

#((station,date),max,min)
daily_avg = daily_max.join(daily_min)
#daily average: (station,date),average_temperature)
daily_avg = daily_avg.map(lambda x: (x[0],((x[1][0]+x[1][1])/2)))

#monthly temperature
#((station,year,month),(avg,1))
month = daily_avg.map(lambda x:((x[0][0],x[0][1][0:4],x[0][1][5:7]),(x[1],1)))
#count the number of days and temperature:
↪  ((station,year,month),(sum_temp,sum_dates))
month_sum =month.reduceByKey(lambda a,b : (a[0]+b[0],a[1]+b[1]))
#average temp of a month
month_avg = month_sum.map(lambda x:
↪  ((x[0][1],x[0][2],x[0][0]),round(x[1][0]/x[1][1],2)))


print("Average temperature:")
print(month_avg.take(10))

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder
month_avg.saveAsTextFile("BDA/output/exercise3")
```

- The average monthly temperature for each available station. *(partial results)*

```
[
(('1964', '03', '102190'), -2.13), (('1964', '05', '102190'), 11.65),
(('1964', '07', '102190'), 15.04), (('1965', '02', '102190'), -4.21),
(('1972', '01', '102190'), -8.34), (('1972', '08', '102190'), 14.13),
(('1972', '11', '102190'), 0.24), (('1975', '02', '102190'), -5.57),
(('1979', '06', '102190'), 14.3), (('2002', '10', '102190'), 0.67)
]
```

4 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:
Station number, maximum measured temperature, maximum daily precipitation

- code

```python
#!/usr/bin/env python3

from pyspark import SparkContext

#step 1 : create a SparkContext which tells Spark how to access a cluster
sc = SparkContext(appName = "exercise 1")

#step 2 : create distributed datasets
# This path is to the file on hdfs
temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
tem_lines = temperature_file.map(lambda line: line.split(";"))

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

#step 3 : RDD operation

#1.temperature:
#(key, value) = (station,temperature)
#(102170,6.3)
temperature = tem_lines.map(lambda x: ((x[0]), float(x[3])))
#get maximum temperature for stations
tem_max = temperature.reduceByKey(lambda a,b: max(a,b))
print("Top 10 max temp:")
print(tem_max.take(10))
#filter: temp[25,30]
tem_max_filter = tem_max.filter(lambda x: (x[1])>=25 and (x[1])<=30)
```

```
#2.precipitation:
#(key, value) = (station-date,precipitation)
#((103100-1995-08-01),0.0)
precipitation=pre_lines.map(lambda x: ((x[0],x[1]), float(x[3])))

#compute daily precipitation
precipitation=precipitation.reduceByKey(lambda a,b: a+b)
#converte to (station, precipitation)
precipitation=precipitation.map(lambda x : (x[0][0],x[1]))

#get maximum precipitation for stations
pre_max = precipitation.reduceByKey(lambda a,b: max(a,b))
print("Top 10 max precipitation:")
print(pre_max.take(10))
#filter: precipitation[100,200]
pre_max_filter =pre_max.filter(lambda x: (x[1])>=100 and (x[1])<=200)

joined_result=tem_max_filter.join(pre_max_filter)


print("Maximum temperature and precipitation:")
print(joined_result.take(10))

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder
joined_result.saveAsTextFile("BDA/output/exercise4")
```

No results were returned for the query filtering stations with a maximum temperature between 25°C and 30°C and a maximum daily precipitation between 100 mm and 200 mm. This suggests that no stations in the dataset satisfy both criteria.

5 ASSIGNMENT 5

Calculate the average monthly precipitation for the Östergotland region (list of stations is provided in a separate file) for the period 1993-2016. In orderto dothis, you willfirstneed 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

- code

```
#!/usr/bin/env python3

from pyspark import SparkContext

#step 1 : create a SparkContext which tells Spark how to access a cluster
sc = SparkContext(appName = "exercise 1")

#step 2 : create distributed datasets
# This path is to the file on hdfs
stations_file = sc.textFile("BDA/input/stations-Ostergotland.csv")
stations_lines = stations_file.map(lambda line: line.split(";"))
stations_lines = stations_lines.map(lambda x: x[0])         #(station,1)
stations_lines = stations_lines.collect()
#set and broadcast stations-Ostergotland.csv
stations_set = set(stations_lines)
broadcast_stations = sc.broadcast(stations_set)


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

#filter precipitation
#stations belong to Ostergotland
pre_filtered = pre_lines.filter(lambda x : x[0] in broadcast_stations.value)
pre_filtered = pre_filtered.map(lambda x:
↪  ((x[0],x[1][0:4],x[1][5:7]),float(x[3])))  #[(103100,1995,08),0.0]
#years within [1993-2016]
pre_filtered = pre_filtered.filter(lambda x: 1993<=int(x[0][1])<=2016)


#sum of monthly precipitation
stations_monthlyPre = pre_filtered.reduceByKey(lambda a,b:a+b)
↪  #[(station,year,month),StationMonthPrecipitation]
#convert to [(year,month),StationMonthPrecipitation] ,each Key-value pair
↪  represent one station
stations_monthlyPre = stations_monthlyPre.map(lambda x: ((x[0][1],x[0][2]),x[1]))
↪  #[(year,month),StationMonthPrecipitation]

# compute total precipitation per month and sum up the number of stations

zeroValue = (0.0,0)         #initial value(total precipitation:0,num of
↪  stations:0), acc
seq_func = lambda acc,val: (acc[0]+val,acc[1]+1)   #acc:(total
↪  precipitation:0,num of stations:0),val:the new precipitation to be aggregated
comb_func = lambda acc1,acc2: (acc1[0]+acc2[0],acc1[1]+acc2[1])
↪  #(sum1,count1),(sum2,count2) sum up
sum_count_perMonth = stations_monthlyPre.aggregateByKey(
    zeroValue,
    seq_func,
    comb_func
)

average_prec_permonth =sum_count_perMonth.mapValues(lambda x: x[0]/x[1])
average_prec_permonth = average_prec_permonth.sortByKey()
```

```
print("average precipitation:")
print(average_prec_permonth.take(10))

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder
average_prec_permonth.saveAsTextFile("BDA/output/exercise5")
```

- The the average monthly precipitation for the Östergotland region for the period 1993-2016*(partial results)*

```
[(('1993', '04'), 0.0), (('1993', '05'), 21.100000000000005),
(('1993', '06'), 56.5), (('1993', '07'), 95.39999999999999),
(('1993', '08'), 80.70000000000005), (('1993', '09'), 40.6),
(('1993', '10'), 43.2), (('1993', '11'), 42.80000000000003),
(('1993', '12'), 37.10000000000003), (('1994', '01'), 22.099999999999994)]
```
