LAB-1

Siddhesh Sreedar (sidsr770)
Hugo Morvan (hugmo418)

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 Please notice that filtering before the reduce step will save the time and resource for running your program

Code :

```
#!/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
max_temperatures = year_temperature.reduceByKey(max)
min_temperatures = year_temperature.reduceByKey(min)

print(min_temperatures.collect())

temperatures_join = max_temperatures.join(min_temperatures)
```

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

#print(max_temperatures.collect())

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

Output:

```
================= FINAL OUTPUT
=========================================
('1975', (36.1, -37.0))
('1992', (35.4, -36.1))
('1994', (34.7, -40.5))
('2010', (34.4, -41.7))
('2014', (34.4, -42.5))
('1989', (33.9, -38.2))
('1982', (33.8, -42.2))
('1968', (33.7, -42.0))
('1966', (33.5, -49.4))
('1983', (33.3, -38.2))
('2002', (33.3, -42.2))
('1970', (33.2, -39.6))
('1986', (33.2, -44.2))
('2000', (33.0, -37.6))
('1956', (33.0, -45.0))
('1959', (32.8, -43.6))
('1991', (32.7, -39.3))
('2006', (32.7, -40.6))
('1988', (32.6, -39.9))
('2011', (32.5, -42.0))
('1999', (32.4, -49.0))
('1953', (32.2, -38.4))
('1973', (32.2, -39.3))
('2008', (32.2, -39.3))
('2007', (32.2, -40.7))
('1955', (32.2, -41.2))
('2003', (32.2, -41.5))
('2005', (32.1, -39.4))
('1969', (32.0, -41.5))
('1979', (32.0, -44.0))
('2001', (31.9, -44.0))
```

('1997', (31.8, -40.2))
('1977', (31.8, -42.5))
('2013', (31.6, -40.7))
('2009', (31.5, -38.5))

Question 2

Count the number of readings for each month in the period of 1950-2014 which are higher than 10 degrees.
The output should contain the following information: Year, month, count

Code:

```
#!/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-month,station),temperature)
year_temperature = lines.map(lambda x: ((x[1][0:7],x[0]),
float(x[3])))

#filter
year_temperature = year_temperature.filter(lambda x:
int(x[0][0][0:4])>=1950 and int(x[0][0][0:4])<=2014)
year_temperature_10 = year_temperature.filter(lambda x:
float(x[1])>10)

#map
year_temperature_group=year_temperature_10.map(lambda x:
(x[0][0],1))

#year_temperature_group_2=year_temperature_group.map(lambda x: ((x[

#Get count
counts = year_temperature_group.reduceByKey(lambda x1,x2: x1 + x2)

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

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

Output:

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

('1958-01', 1)
('1981-05', 35371)
(...)

Part-2

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.

Code:

```
#!/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-month-station),temperature)
year_temperature = lines.map(lambda x: ((x[1][0:7]+x[0],x[0]),
float(x[3])))

#filter year and temperature
year_temperature = year_temperature.filter(lambda x:
int(x[0][0][0:4])>=1950 and int(x[0][0][0:4])<=2014)
year_temperature_10 = year_temperature.filter(lambda x:
float(x[1])>10)

#reduce by month-station pair and reset count
unique_month_station = year_temperature_10.reduceByKey(lambda x1,x2:
x1)
no_dup_station_count = unique_month_station.map(lambda x:
(x[0][0],1))

#remove station id
month_map = no_dup_station_count.map(lambda x: (x[0][0:7],x[1]))

#Get count
counts = month_map.reduceByKey(lambda x1,x2: x1 + x2)
```

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

Output:

```
================= FINAL OUTPUT ======================================
('2000-08', 325)
('2008-10', 226)
('1996-06', 345)
('1983-11', 160)
('2013-09', 299)
('1994-05', 299)
('1990-02', 148)
('1970-10', 345)
('2010-03', 65)
('1961-03', 197)
('1990-09', 312)
('2003-05', 321)
('1988-06', 322)
('1995-03', 59)
('1953-03', 77)
('2010-08', 318)
('1962-06', 297)
('1983-07', 319)
('1990-03', 193)
('1957-06', 128)
('1965-06', 355)
('1989-09', 316)
('1969-09', 359)
('1963-04', 283)
('1995-09', 315)
('1959-04', 115)
('1982-04', 246)
('1967-10', 324)
('1951-09', 112)
('1992-04', 181)
('2001-10', 279)
('1953-04', 104)
('1995-11', 24)
('1954-11', 21)
('1974-11', 19)
('1963-03', 1)
('1958-01', 1)
('1983-03', 17)
('1991-01', 1)
```

('1952-11', 1)
('1950-12', 1)
('1962-03', 1)
('1960-01', 1)
('1967-07', 351)
('2005-08', 306)
('1987-04', 261)
('2010-05', 319)
('1967-06', 359)
('1998-09', 326)
('1968-04', 322)
('1956-10', 103)
(...)

Question 3

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

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_temperature = lines.map(lambda x: (x[1][0:7]+"-"+x[0],
float(x[3])))

#filter
year_temperature = year_temperature.filter(lambda x:
int(x[0][0:4])>=1960 and int(x[0][0:4])<=2014)

year_temperature_map=year_temperature.map(lambda x: (x[0],(x[1],1)))
```

```
year_temperature_sum= year_temperature_map.reduceByKey(lambda x,y:
(x[0]+ y[0],x[1]+y[1]))

monthly_average = year_temperature_sum.map(lambda x
:(x[0],x[1][0]/x[1][1]))

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

Output:

========== FINAL OUTPUT
==========
('1993-07-140360', 14.286693548387097)
('1994-07-140360', 16.391532258064508)
('1994-11-140360', 0.9116666666666666)
('1996-03-140360', -3.82540322580645)
('1999-03-140360', -3.135080645161292)
('2000-04-140360', 1.7020833333333343)
('2002-08-140360', 18.6532258064516)
('2002-10-140360', 2.4834677419354843)
('2003-02-140360', -3.8409972299168977)
('2005-12-140360', -1.7034946236559125)
('2006-03-140360', -7.706451612903231)
('2007-08-140360', 15.258198924731149)
('2008-01-140360', -2.8551020408163286)
('2005-06-140460', 11.51838440111421)
('2006-08-140460', 18.732972972972977)
('2006-11-140460', 1.8534023668639064)
('2007-05-140460', 6.557412398921838)
('2008-04-140460', 2.056527777777778)
('2009-02-140460', -8.056184798807749)
('2010-02-140460', -12.296428571428574)
('2010-08-140460', 14.17459677419355)
('2011-05-140460', 6.694623655913981)
('2011-12-140460', 1.1005376344086026)
('2012-07-140460', 15.567876344086022)
('2013-04-140460', 0.32972222222222175)
('1962-03-140480', -9.41140776699029)
('1962-10-140480', 5.789978213507626)
('1965-07-140480', 14.141263440860211)
('1966-10-140480', 2.7021505376344086)
('1967-10-140480', 4.533870967741937)
('1968-08-140480', 13.883198924731186)
('1969-02-140480', -13.652232142857141)
('1971-06-140480', 12.365277777777775)

('1972-02-140480', -7.033908045977014)
('1975-01-140480', -4.68700410396717)
('1975-04-140480', 1.7612500000000004)
('1977-11-140480', -1.8829166666666635)
('1978-07-140480', 13.541160593792158)
('1978-12-140480', -13.575552486187851)
('1979-11-140480', -1.0391666666666666)
('1979-12-140480', -6.043888888888897)
('1980-09-140480', 10.319888734353277)
('1980-12-140480', -7.776290097628999)
('1981-10-140480', 3.2803763440860183)
('1982-02-140480', -6.499107142857148)
('1982-08-140480', 14.069272237196765)
('1984-01-140480', -9.923315363881411)
('1984-05-140480', 10.494885598923288)
('1984-07-140480', 15.131664411366716)

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

Code:

```
#!/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")
prec_file = sc.textFile("BDA/input/precipitation-readings.csv")
lines = temperature_file.map(lambda line: line.split(";"))
lines2 = prec_file.map(lambda line: line.split(";"))


#temperature filtering
temp_keyed = lines.map(lambda x: (x[0], float(x[3]) ))
temp_max = temp_keyed.reduceByKey(max)
temp_filter = temp_max.filter(lambda x: x[1]>= 25  and x[1]<=30)
```

```
#precipitation filtering
prec_keyed = lines2.map(lambda x: (x[0]+x[1], float(x[3]) ))
prec_daily = prec_keyed.reduceByKey(lambda x,y: x + y)
prec_max = prec_daily.reduceByKey(max)
prec_filter = prec_max.filter(lambda x: x[1]>= 100 and x[1]<= 200)
no_date = prec_filter.map(lambda x: (x[0][0:6],x[1]) )

#join the two filtered dataset
table_join = temp_filter.join(no_date)

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

Output:

========== FINAL OUTPUT ==========

==========

(No output, no station met those conditions)

Sanity check:

Temperature output after filtering:

========== FINAL OUTPUT ==========

('117330', 26.2)
('123060', 28.0)
('162880', 29.9)
('81220', 30.0)
('132180', 27.8)
('133050', 27.3)
('177930', 26.5)
('183750', 29.4)
('188800', 27.9)
('82030', 30.0)
('82250', 27.6)
('106630', 28.6)
('155910', 28.1)
('158740', 29.2)
('158850', 28.7)
('71360', 27.0)
('99450', 26.0)
('147560', 29.9)
('151220', 29.5)
('151550', 29.8)
('155710', 26.8)

('133180', 30.0)
('177920', 26.2)
('179950', 28.0)
('180770', 28.9)
('182910', 29.2)
('81620', 25.0)
('82360', 29.9)
('84260', 29.1)
('84660', 27.6)
('137560', 29.8)
('139340', 28.9)
('86360', 27.5)
('114630', 28.0)
('116230', 27.6)
('162800', 25.5)
('162970', 30.0)
('96370', 30.0)
('135380', 28.0)
('135640', 25.6)
('139570', 28.0)
('191910', 27.7)
('52230', 29.6)
(…)

Precipitation output after filtering:
========== FINAL OUTPUT ==========
('714201', 106.3)
('752502', 101.8)
('523502', 101.6)
('975101', 103.99999999999999)
==========

It can be assumed that there is no station in common between the two tables before the join, thus the empty output

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 order to do this, 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

Code:

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

from pyspark import SparkContext

sc = SparkContext(appName = "exercise 1")
# This path is to the file on hdfs
prec_file = sc.textFile("BDA/input/precipitation-readings.csv")
lines = prec_file.map(lambda line: line.split(";"))

#(key, value) = ((year-month-station), precipitation)
lines = lines.map(lambda x: (x[1][0:7]+"-"+x[0], float(x[3])))
filtered_period = lines.filter(lambda x: int(x[0][0:4])>=1993 and
int(x[0][0:4])<=2016)

#getting the average monthly precipitation per station for all the
stations
sum_per_station_per_month = filtered_period.reduceByKey(lambda x, y:
(x+y))

#filtering to keep only the stations in Ostergotland:
#load the stations
stations_file = sc.textFile("BDA/input/stations-Ostergotland.csv")
stations = stations_file.map(lambda line: line.split(";"))
stations = stations.map(lambda x: x[0]).collect()
#filter to keep only stations in stations list
filtered = sum_per_station_per_month.filter(lambda x: x[0][8:13] in
stations)

#Averaging over the stations
sum_count=filtered.map(lambda x: (x[0][0:7],(x[1],1)))
sum_count= sum_count.reduceByKey(lambda x,y: (x[0]+ y[0],x[1]+y[1]))
monthly_average = sum_count.map(lambda x :(x[0],x[1][0]/x[1][1]))

#result = monthly_average

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

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

Output:

```
================= FINAL OUTPUT
=======================================
('1997-04', 25.950000000000006)
('1997-06', 86.98333333333335)
('1998-03', 33.90000000000003)
('2005-07', 104.34999999999998)
('2005-10', 38.05000000000001)
('2009-08', 61.566666666666684)
('2010-05', 67.1666666666667)
('2012-06', 132.19999999999996)
('2014-10', 72.13749999999999)
('2014-12', 35.46250000000001)
('2016-04', 26.900000000000006)
('2016-05', 29.250000000000004)
('1999-11', 18.450000000000003)
('2006-04', 44.36666666666668)
('2011-05', 37.85)
('2013-08', 54.07500000000001)
('1999-08', 54.80000000000002)
('1999-10', 18.549999999999997)
('2000-05', 25.316666666666677)
('2002-10', 60.50000000000002)
('2008-04', 20.25)
('2009-12', 53.450000000000045)
('1996-10', 22.45)
('2007-02', 33.06666666666667)
('2007-10', 28.116666666666674)
('2000-09', 27.51666666666668)
('2004-10', 78.18333333333332)
('2012-05', 22.96666666666667)
('2012-12', 66.9333333333334)
('2013-07', 54.5625)
('2008-05', 23.133333333333336)
('2013-12', 42.262500000000024)
('2015-04', 15.337499999999999)
('2008-09', 47.366666666666696)
('2012-10', 65.58333333333333)
('1993-04', 0.0)
('1995-06', 97.19999999999987)
('1998-05', 38.366666666666696)
('2014-09', 48.45000000000002)
('2005-08', 76.96666666666667)
```

('2003-02', 9.116666666666665)
('2009-05', 54.166666666666686)
('2012-09', 72.75)

Q.3) **Fix**

##################################
Code:

```
#!/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-month-day-station),temperature)
year_temperature = lines.map(lambda x: (x[1][0:10]+"-"+x[0], float(x[3])))

#filter
year_temperature = year_temperature.filter(lambda x: int(x[0][0:4])>=1960 and
int(x[0][0:4])<=2014)

max_temperatures = year_temperature.reduceByKey(max)
min_temperatures = year_temperature.reduceByKey(min)

temperatures_join = max_temperatures.join(min_temperatures)


daily_average=temperatures_join.map(lambda x: (x[0],((x[1][0]+x[1][1])/2)))
#removing the day from the key
year_temperature = daily_average.map(lambda x: (x[0][0:7]+x[0][-7:], x[1]))

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

year_temperature_sum= year_temperature_map.reduceByKey(lambda x,y: (x[0]+
y[0],x[1]+y[1]))

monthly_average = year_temperature_sum.map(lambda x :(x[0],x[1][0]/x[1][1]))

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

Output:

========== FINAL OUTPUT ==========

('1960-12-102190', -2.5064516129032253)
('1965-03-102190', -2.7129032258064516)
('1965-05-102190', 8.751612903225805)
('1966-06-102190', 17.438333333333336)
('1966-11-102190', 0.22000000000000006)
('1968-06-102190', 16.681666666666665)
('1971-10-102190', 5.809677419354839)
('1973-10-102190', 1.4564516129032257)
('1975-03-102190', -0.8048387096774193)
('1975-07-102190', 15.87741935483871)
('1975-11-102190', 1.1066666666666667)
('1977-10-102190', 5.741935483870968)
('1978-02-102190', -9.798214285714286)
('2002-05-102190', 12.506451612903227)
('2004-02-102190', -5.210344827586208)
('2004-12-102190', -3.0790322580645157)
('2005-01-102190', -0.3741935483870967)
('2005-12-102190', -4.256451612903226)
('2009-12-102190', -7.140322580645161)
('1980-10-102200', 2.0403225806451615)
('1981-02-102200', -6.453571428571428)
('1982-08-102200', 14.26774193548387)
('1982-11-102200', 0.7483333333333334)
('1986-08-102200', 11.16451612903226)
('1986-09-102200', 6.258333333333335)
('1988-01-102200', -0.9935483870967743)
('1988-12-102210', -5.619354838709677)
('1989-01-102210', 0.8983870967741936)
('1990-04-102210', 3.9883333333333337)
('1990-05-102210', 9.503225806451612)
('1992-10-102210', 0.6145161290322582)
('1993-08-102210', 10.611290322580645)
('1994-03-102210', -1.9370967741935485)
('1996-08-102210', 15.106451612903225)
('1997-09-102390', 7.21)
('2000-05-102390', 9.290322580645162)
('2001-07-102390', 15.427419354838712)
('2003-11-102390', 0.58)
('2005-07-102390', 16.193548387096772)
('2007-06-102390', 14.06)
('2007-08-102390', 14.066129032258063)
('2008-02-102390', -1.2448275862068967)
('2011-12-102390', -2.9483870967741934)