lab2

Zhixuan_Duan(zhidu838)

2020_07_04

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.

sparkSQL code

```
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row, functions

data1 = "/Users/darin/Desktop/Big_Data/data/temperature-readings.csv"
result1 = '/Users/darin/Desktop/Big_Data/data/sql_max_temperature'
result2 = '/Users/darin/Desktop/Big_Data/data/sql_min_temperature'
startYear = 1950
endYear = 2014

sc = SparkContext(appName = "lab2-1")

lines = sc.textFile(data1)
lines01 = lines.map(lambda x: x.split(";"))
lines02 = lines01.filter(lambda x: int(x[1][0:4]) >= startYear and
                                   int(x[1][0:4]) <= endYear)
dataTemp = lines02.map(lambda x: Row(station=x[0], date=x[1], year=x[1].split("-")[0], time=x[2],temp=fl

dataTemp.registerTempTable("lab2-1-df")

# max_temp
maxTemp = dataTemp1.groupBy("year").agg(functions.max("temp").alias("temp")) \
    .orderBy(["temp"], ascending = [0])

maxTemp = maxTemp.join(dataTemp1.select("station","year","temp"),["year","temp"],'left_outer')
maxTemp = maxTemp.orderBy(["temp"], ascending=[0])
maxTemp = maxTemp.rdd.sortBy(keyfunc=lambda x: x[1], ascending=False)
maxTemp.saveAsTextFile(result1)

## min_temp
minTemp = dataTemp1.groupBy("year").agg(functions.min("temp").alias("temp")) \
    .orderBy(["temp"], ascending = [0])
minTemp = minTemp.join(dataTemp1.select("station","year","temp"),["year","temp"],'left_outer')
minTemp = minTemp.orderBy(["temp"], ascending=[0])
minTemp = minTemp.rdd.sortBy(keyfunc=lambda x: x[1], ascending=False)
minTemp.saveAsTextFile(result2)
```

sql_max_temp

```
year temp station
1975 36.1 86200
1992 35.4 63600
1994 34.7 117160
2014 34.4 96560
2010 34.4 75250
1989 33.9 63050
1982 33.8 94050
1968 33.7 137100
1966 33.5 151640
1983 33.3 98210
...
1987 29.6 96310
1984 29.5 105370
1967 29.5 75040
1960 29.4 173810
1950 29.4 75040
1998 29.2 63600
1965 28.5 116500
1951 28.5 75040
1962 27.4 76380
1962 27.4 86200
```

sql_min_temp

```
year  temp station
1990 -35.0 147270
1990 -35.0 166870
1952 -35.5 192830
1974 -35.6 166870
1974 -35.6 179950
1954 -36.0 113410
1992 -36.1 179960
1975 -37.0 157860
1972 -37.5 167860
1995 -37.6 182910
...
1971 -44.3 166870
1956 -45.0 160790
1980 -45.0 191900
1980 -45.0 191900
1967 -45.4 166870
1987 -47.3 123480
1978 -47.7 155940
1999 -49.0 192830
1999 -49.0 192830
1966 -49.4 179950
```

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

sparkSQL code

```
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row, functions

data1 = "/Users/darin/Desktop/Big_Data/data/temperature-readings.csv"
result2_1 = '/Users/darin/Desktop/Big_Data/data/sql_over_ten_temp_counts'
result2_2 = '/Users/darin/Desktop/Big_Data/data/sql_over_ten_temp_distinct_counts'
startYear = 1950
endYear = 2014

sc = SparkContext(appName = "lab2-2")

lines = sc.textFile(data1)
lines01 = lines.map(lambda x: x.split(";"))
lines02 = lines01.filter(lambda x: int(x[1][0:4]) >= startYear and
                                   int(x[1][0:4]) <= endYear)
dataTemp = lines02.map(lambda x: Row(station=x[0], date=x[1], year=x[1].split("-")[0],
                                     month = x[1].split("-")[1],time=x[2],
                                     temp=float(x[3]), quality=x[4]))

dataTemp.registerTempTable("lab2-2-df")

# 1. year, month, counts
ten_temp = dataTemp.filter(dataTemp["temp"] > 10).groupby(["year","month"]) \
    .agg(functions.count("station"))
ten_temp = ten_temp.sortBy(keyfunc=lambda x:x[2], ascending=False)
ten_temp.saveAsTextFile(result2_1)

# 2. year, month, distinct counts
distinct_ten_temp = dataTemp.filter(dataTemp["temp"] > 10).groupby(["year","month"]) \
    .agg(functions.countDistinct("station"))
distinct_ten_temp = distinct_ten_temp.sortBy(keyfunc=lambda x:x[2], ascending=False)
distinct_ten_temp.saveAsTextFile(result2_2)
```

over_ten_mth_temp_counts

```
year-month counts
1997-04 3756
1974-07 66277
2003-05 48264
1981-10 9882
```

```
1981-03 395
1987-05 17191
2009-07 133008
1986-11 1198
1983-09 38692
1952-03 1
...

2013-10 37839
1973-05 27886
1960-12 2
1972-11 993
1959-05 12445
1971-10 13326
1960-05 14333
1966-07 55544
1976-02 19
1983-06 49265
```

over_ten_temp_distinct_counts

```
year-month counts
1997-04 91130
1974-07 77220
2003-05 182930
1981-10 91130
1983-09 157860
1987-05 122330
1979-04 74180
2009-07 106040
1986-11 81540
1966-08 72120
...

2013-10 136360
1973-05 105450
1960-12 134110
1972-11 65360
1959-05 135440
1971-10 65360
1960-05 97620
1966-07 85160
1976-02 75040
1955-01 84310
```

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

sparkSQL code

```
from pyspark import SparkContext
from pyspark.sql import Row, functions

data1 = "/Users/darin/Desktop/Big_Data/data/temperature-readings.csv"
result3 = "/Users/darin/Desktop/Big_Data/data/avg_month_temp"
startYear = 1960
endYear = 2014

sc = SparkContext(appName="lab2-3")

lines = sc.textFile(data1)
lines01 = lines.map(lambda x: x.split(";"))
lines02 = lines01.filter(lambda x: int(x[1][0:4]) >= startYear and
                                   int(x[1][0:4]) <= endYear)
dataTemp = lines02.map(lambda x: Row(station=x[0], date=x[1], year=x[1].split("-")[0],
                                     month = x[1].split("-")[1],time=x[2],
                                     temp=float(x[3]), quality=x[4]))

dataTemp.registerTempTable("lab2-3-df")

# get the avg daily temp
minTemp = dataTemp.groupBy("year", "month", "day", "station") \
    .agg(functions.min("temp").alias("minT"))
maxTemp = dataTemp.groupBy("year", "month", "day", "station") \
    .agg(functions.max("temp").alias("maxT"))

min_maxTemp = minTemp.join(maxTemp,["year","month","day","station"])
min_maxTemp = min_maxTemp.withColumn("avg_temp",(min_maxTemp.minT + min_maxTemp.maxT)/2) \.select("year

# get the monthly temp
avg_month_temp = min_maxTemp.groupBy("year","month","station")\
    .agg(functions.avg("avg_temp")).alias("avg_month_temp") \
    .orderBy(["avg_month_temp"], ascending = [0])

avg_month_temp = avg_month_temp.sortBy(keyfunc=lambda x:x[3], ascending=False)
avg_month_temp.saveAsTextFile(result3)
```

avg_station_month_temp

```
year-month station temp
2001-01 63280 0.0032258064516128343
1969-03 83620 -4.780645161290323
1978-09 156730 5.5566666666666675
1991-11 106500 -0.05166666666666667
1995-06 112080 12.001666666666667
1998-05 172770 5.282258064516129
2003-06 177930 8.711666666666668
1978-09 95160 10.040000000000001
1976-10 89240 6.138709677419354
2001-02 159770 -14.35
...
1976-08 54400 16.81129032258065
1990-05 181900 6.77258064516129
1964-12 87150 0.04516129032258059
2003-08 134590 12.309677419354841
1965-04 158750 1.6566666666666665
1972-07 94140 19.853225806451615
2002-10 83440 4.762903225806452
1970-12 163920 -5.527419354838709
1965-01 74470 -0.7516129032258065
1983-12 161780 -8.535483870967743
```

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 100mm 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

sparkSQL code

```
from pyspark import SparkContext
from pyspark.sql import Row, functions

data1 = "/Users/darin/Desktop/Big_Data/data/temperature-readings.csv"
data2 = "/Users/darin/Desktop/Big_Data/data/precipitation-readings.csv"
result4 = "/Users/darin/Desktop/Big_Data/data/max_temp_prec"

sc = SparkContext(appName="lab2-4")

# data temp
dataTemp = sc.textFile(data1)
dataTemp = dataTemp.map(lambda x: x.split(";"))
dataTemp = dataTemp.filter(lambda x: int(x[1][0:4]) >= 1950 and
                                     int(x[1][0:4]) <= 2014)
dataTemp = dataTemp.map(lambda x: Row(station=x[0], temp=float(x[3])))

dataTemp = dataTemp.registerTempTable("lab2-4-temp-df")

# get the max-temp
maxTemp = dataTemp.groupBy("station").agg(functions.max("temp").alias("maxTemp"))
maxTemp = maxTemp.filter(maxTemp.maxTemp > 25 and
                         maxTemp.maxTemp < 30)

# data prec
dataPrec = sc.textFile(data2)
dataPrec = dataPrec.map(lambda x: x.split(";"))
dataPrec = dataPrec.filter(lambda x: int(x[1][0:4]) >= 1950 and
                                     int(x[1][0:4]) <= 2014)
dataPrec = dataPrec.map(lambda x: Row(station=x[0],prec=float(x[3])))

dataPrec = dataPrec.registerTempTable("lab2-4-prec-df")

# get the max-prec
maxPrec = dataPrec.groupBy("station").agg(functions.max("prec").alias("maxPrec"))
maxPrec = maxPrec.filter(maxPrec.maxPrec > 100 and
                         maxPrec.maxPrec < 200)

# merge them together
maxTempPrec = maxTemp.join(maxPrec, "station")
maxTempPrec = maxTempPrec.orderby(["station"], ascending = [0])
maxTempPrec.saveAsTextFile(result4)
```

max_temp_prec

```
station temp prec
99280 25.5 34.00000000000001
87440 26.2 45.1
82360 29.9 84.30000000000001
78550 29.9 49.199999999999996
71190 28.3 80.3999999999999
68560 28.5 52.99999999999999
66110 26.3 72.60000000000002
52240 29.0 79.6
188790 26.6 34.199999999999996
183750 29.4 38.1
...
158740 29.2 61.1
157870 29.9 61.29999999999999
144310 28.4 53.5
139260 29.4 86.30000000000001
139120 27.9 95.4
135460 29.2 62.1
122260 28.5 47.6
178860 27.4 44.900000000000006
133500 27.2 50.10000000000001
132170 28.0 39.6
```

Assignment 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 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

sparkSQL code

```
from pyspark import SparkContext
from pyspark.sql import Row, functions

data1 = "/Users/darin/Desktop/Big_Data/data/precipitation-readings.csv"
data2 = "/Users/darin/Desktop/Big_Data/data/stations-Ostergotland.csv"
result5 = "/Users/darin/Desktop/Big_Data/data/avg_prec_ost"

sc = SparkContext(appName="lab2-5")

# get data and pre-process
station = sc.textFile(data1).cache()
station = station.map(lambda line:line.split(";"))
station = station.map(lambda x:Row(station=x[0], name=x[1]))
stations = sqlContext.createDataFrame(station)
stations.registerTempTable("q5_Stations")

dataPrec = sc.textFile(data2).cache()
dataPrec = dataPrec.map(lambda line:line.split(";"))
dataPrec = dataPrec.filter(lambda x:(int(x[1][0:4])>=1993 and int(x[1][0:4])<=2016)) dataPrec = dataPre
time=x[2], prec=float(x[3]), quality=x[4]))
dataPrec = sqlContext.createDataFrame(dataPrec)
dataPrec.registerTempTable("q5_dataPrec")

# merge together and get the avg_prec
Result = stations.join(dataPrec, "station")
Result = Result.groupBy("year", "month", "station").agg(F.sum("prec").alias("precSum")) Result = Result
Result = Result.orderBy(["year", "month"], ascending=[0,0])
Result.saveAsTextFile("result5")
```

avg_prec_ost

```
year month prec
2016 05 29.250000000000007
2016 02 21.5625
2015 12 28.925
2015 11 63.88750000000002
```

```
2015 09 101.29999999999998
2015 08 26.9875
2015 07 119.09999999999995
2015 06 78.66250000000001
2015 05 93.225
2015 04 15.3375
...
2013 07 54.56249999999999
2013 05 47.92500000000001
2013 04 38.287500000000016
2013 02 25.52500000000002
2012 11 68.65
2012 10 65.58333333333334
2012 09 72.75
2012 07 59.06666666666667
2012 04 62.78333333333335
2012 01 43.55000000000003
```

Appendix

```
knitr::opts_chunk$set(echo = TRUE,
                      eval = FALSE,
                      warning = FALSE,
                      message = FALSE,
                      fig.width = 7,
                      fig.height = 4,
                      fig.align = 'center')
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row, functions

data1 = "/Users/darin/Desktop/Big_Data/data/temperature-readings.csv"
result1 = '/Users/darin/Desktop/Big_Data/data/sql_max_temperature'
result2 = '/Users/darin/Desktop/Big_Data/data/sql_min_temperature'
startYear = 1950
endYear = 2014

sc = SparkContext(appName = "lab2-1")

lines = sc.textFile(data1)
lines01 = lines.map(lambda x: x.split(";"))
lines02 = lines01.filter(lambda x: int(x[1][0:4]) >= startYear and
                                   int(x[1][0:4]) <= endYear)
dataTemp = lines02.map(lambda x: Row(station=x[0], date=x[1], year=x[1].split("-")[0], time=x[2],temp=fl

dataTemp.registerTempTable("lab2-1-df")

# max_temp
maxTemp = dataTemp1.groupBy("year").agg(functions.max("temp").alias("temp")) \
    .orderBy(["temp"], ascending = [0])

maxTemp = maxTemp.join(dataTemp1.select("station","year","temp"),["year","temp"],'left_outer')
maxTemp = maxTemp.orderBy(["temp"], ascending=[0])
maxTemp = maxTemp.rdd.sortBy(keyfunc=lambda x: x[1], ascending=False)
maxTemp.saveAsTextFile(result1)

## min_temp
minTemp = dataTemp1.groupBy("year").agg(functions.min("temp").alias("temp")) \
    .orderBy(["temp"], ascending = [0])
minTemp = minTemp.join(dataTemp1.select("station","year","temp"),["year","temp"],'left_outer')
minTemp = minTemp.orderBy(["temp"], ascending=[0])
minTemp = minTemp.rdd.sortBy(keyfunc=lambda x: x[1], ascending=False)
minTemp.saveAsTextFile(result2)
year temp station
1975 36.1 86200
1992 35.4 63600
1994 34.7 117160
2014 34.4 96560
2010 34.4 75250
1989 33.9 63050
1982 33.8 94050
1968 33.7 137100
```

```
1966 33.5 151640
1983 33.3 98210
...
1987 29.6 96310
1984 29.5 105370
1967 29.5 75040
1960 29.4 173810
1950 29.4 75040
1998 29.2 63600
1965 28.5 116500
1951 28.5 75040
1962 27.4 76380
1962 27.4 86200
year  temp station
1990 -35.0 147270
1990 -35.0 166870
1952 -35.5 192830
1974 -35.6 166870
1974 -35.6 179950
1954 -36.0 113410
1992 -36.1 179960
1975 -37.0 157860
1972 -37.5 167860
1995 -37.6 182910
...
1971 -44.3 166870
1956 -45.0 160790
1980 -45.0 191900
1980 -45.0 191900
1967 -45.4 166870
1987 -47.3 123480
1978 -47.7 155940
1999 -49.0 192830
1999 -49.0 192830
1966 -49.4 179950
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row, functions

data1 = "/Users/darin/Desktop/Big_Data/data/temperature-readings.csv"
result2_1 = '/Users/darin/Desktop/Big_Data/data/sql_over_ten_temp_counts'
result2_2 = '/Users/darin/Desktop/Big_Data/data/sql_over_ten_temp_distinct_counts'
startYear = 1950
endYear = 2014

sc = SparkContext(appName = "lab2-2")

lines = sc.textFile(data1)
lines01 = lines.map(lambda x: x.split(";"))
lines02 = lines01.filter(lambda x: int(x[1][0:4]) >= startYear and
                                   int(x[1][0:4]) <= endYear)
dataTemp = lines02.map(lambda x: Row(station=x[0], date=x[1], year=x[1].split("-")[0],
                                     month = x[1].split("-")[1],time=x[2],
                                     temp=float(x[3]), quality=x[4]))
```

```
dataTemp.registerTempTable("lab2-2-df")

# 1. year, month, counts
ten_temp = dataTemp.filter(dataTemp["temp"] > 10).groupby(["year","month"]) \
    .agg(functions.count("station"))
ten_temp = ten_temp.sortBy(keyfunc=lambda x:x[2], ascending=False)
ten_temp.saveAsTextFile(result2_1)

# 2. year, month, distinct counts
distinct_ten_temp = dataTemp.filter(dataTemp["temp"] > 10).groupby(["year","month"]) \
    .agg(functions.countDistinct("station"))
distinct_ten_temp = distinct_ten_temp.sortBy(keyfunc=lambda x:x[2], ascending=False)
distinct_ten_temp.saveAsTextFile(result2_2)
year-month counts
1997-04 3756
1974-07 66277
2003-05 48264
1981-10 9882
1981-03 395
1987-05 17191
2009-07 133008
1986-11 1198
1983-09 38692
1952-03 1
...

2013-10 37839
1973-05 27886
1960-12 2
1972-11 993
1959-05 12445
1971-10 13326
1960-05 14333
1966-07 55544
1976-02 19
1983-06 49265
year-month counts
1997-04 91130
1974-07 77220
2003-05 182930
1981-10 91130
1983-09 157860
1987-05 122330
1979-04 74180
2009-07 106040
1986-11 81540
1966-08 72120
...

2013-10 136360
1973-05 105450
1960-12 134110
1972-11 65360
```

```
1959-05 135440
1971-10 65360
1960-05 97620
1966-07 85160
1976-02 75040
1955-01 84310
from pyspark import SparkContext
from pyspark.sql import Row, functions

data1 = "/Users/darin/Desktop/Big_Data/data/temperature-readings.csv"
result3 = "/Users/darin/Desktop/Big_Data/data/avg_month_temp"
startYear = 1960
endYear = 2014

sc = SparkContext(appName="lab2-3")

lines = sc.textFile(data1)
lines01 = lines.map(lambda x: x.split(";"))
lines02 = lines01.filter(lambda x: int(x[1][0:4]) >= startYear and
                                   int(x[1][0:4]) <= endYear)
dataTemp = lines02.map(lambda x: Row(station=x[0], date=x[1], year=x[1].split("-")[0],
                                     month = x[1].split("-")[1],time=x[2],
                                     temp=float(x[3]), quality=x[4]))

dataTemp.registerTempTable("lab2-3-df")

# get the avg daily temp
minTemp = dataTemp.groupBy("year", "month", "day", "station") \
    .agg(functions.min("temp").alias("minT"))
maxTemp = dataTemp.groupBy("year", "month", "day", "station") \
    .agg(functions.max("temp").alias("maxT"))

min_maxTemp = minTemp.join(maxTemp,["year","month","day","station"])
min_maxTemp = min_maxTemp.withColumn("avg_temp",(min_maxTemp.minT + min_maxTemp.maxT)/2) \.select("year

# get the monthly temp
avg_month_temp = min_maxTemp.groupBy("year","month","station")\
    .agg(functions.avg("avg_temp")).alias("avg_month_temp") \
    .orderBy(["avg_month_temp"], ascending = [0])

avg_month_temp = avg_month_temp.sortBy(keyfunc=lambda x:x[3], ascending=False)
avg_month_temp.saveAsTextFile(result3)
year-month station temp
2001-01 63280 0.0032258064516128343
1969-03 83620 -4.780645161290323
1978-09 156730 5.5566666666666675
1991-11 106500 -0.05166666666666667
1995-06 112080 12.001666666666667
1998-05 172770 5.282258064516129
2003-06 177930 8.711666666666668
1978-09 95160 10.040000000000001
1976-10 89240 6.138709677419354
2001-02 159770 -14.35
```

```
...
1976-08 54400 16.81129032258065
1990-05 181900 6.77258064516129
1964-12 87150 0.04516129032258059
2003-08 134590 12.309677419354841
1965-04 158750 1.6566666666666665
1972-07 94140 19.853225806451615
2002-10 83440 4.762903225806452
1970-12 163920 -5.527419354838709
1965-01 74470 -0.7516129032258065
1983-12 161780 -8.535483870967743
from pyspark import SparkContext
from pyspark.sql import Row, functions

data1 = "/Users/darin/Desktop/Big_Data/data/temperature-readings.csv"
data2 = "/Users/darin/Desktop/Big_Data/data/precipitation-readings.csv"
result4 = "/Users/darin/Desktop/Big_Data/data/max_temp_prec"

sc = SparkContext(appName="lab2-4")

# data temp
dataTemp = sc.textFile(data1)
dataTemp = dataTemp.map(lambda x: x.split(";"))
dataTemp = dataTemp.filter(lambda x: int(x[1][0:4]) >= 1950 and
                                     int(x[1][0:4]) <= 2014)
dataTemp = dataTemp.map(lambda x: Row(station=x[0], temp=float(x[3])))

dataTemp = dataTemp.registerTempTable("lab2-4-temp-df")

# get the max-temp
maxTemp = dataTemp.groupBy("station").agg(functions.max("temp").alias("maxTemp"))
maxTemp = maxTemp.filter(maxTemp.maxTemp > 25 and
                         maxTemp.maxTemp < 30)

# data prec
dataPrec = sc.textFile(data2)
dataPrec = dataPrec.map(lambda x: x.split(";"))
dataPrec = dataPrec.filter(lambda x: int(x[1][0:4]) >= 1950 and
                                     int(x[1][0:4]) <= 2014)
dataPrec = dataPrec.map(lambda x: Row(station=x[0],prec=float(x[3])))

dataPrec = dataPrec.registerTempTable("lab2-4-prec-df")

# get the max-prec
maxPrec = dataPrec.groupBy("station").agg(functions.max("prec").alias("maxPrec"))
maxPrec = maxPrec.filter(maxPrec.maxPrec > 100 and
                         maxPrec.maxPrec < 200)

# merge them together
maxTempPrec = maxTemp.join(maxPrec, "station")
maxTempPrec = maxTempPrec.orderby(["station"], ascending = [0])
maxTempPrec.saveAsTextFile(result4)
station temp prec
99280 25.5 34.00000000000001
```

```
87440 26.2 45.1
82360 29.9 84.30000000000001
78550 29.9 49.19999999999996
71190 28.3 80.39999999999999
68560 28.5 52.99999999999999
66110 26.3 72.60000000000002
52240 29.0 79.6
188790 26.6 34.199999999999996
183750 29.4 38.1
...
158740 29.2 61.1
157870 29.9 61.29999999999999
144310 28.4 53.5
139260 29.4 86.30000000000001
139120 27.9 95.4
135460 29.2 62.1
122260 28.5 47.6
178860 27.4 44.900000000000006
133500 27.2 50.10000000000001
132170 28.0 39.6
from pyspark import SparkContext
from pyspark.sql import Row, functions

data1 = "/Users/darin/Desktop/Big_Data/data/precipitation-readings.csv"
data2 = "/Users/darin/Desktop/Big_Data/data/stations-Ostergotland.csv"
result5 = "/Users/darin/Desktop/Big_Data/data/avg_prec_ost"

sc = SparkContext(appName="lab2-5")

# get data and pre-process
station = sc.textFile(data1).cache()
station = station.map(lambda line:line.split(";"))
station = station.map(lambda x:Row(station=x[0], name=x[1]))
stations = sqlContext.createDataFrame(station)
stations.registerTempTable("q5_Stations")

dataPrec = sc.textFile(data2).cache()
dataPrec = dataPrec.map(lambda line:line.split(";"))
dataPrec = dataPrec.filter(lambda x:(int(x[1][0:4])>=1993 and int(x[1][0:4])<=2016)) dataPrec = dataPre
time=x[2], prec=float(x[3]), quality=x[4]))
dataPrec = sqlContext.createDataFrame(dataPrec)
dataPrec.registerTempTable("q5_dataPrec")

# merge together and get the avg_prec
Result = stations.join(dataPrec, "station")
Result = Result.groupBy("year", "month", "station").agg(F.sum("prec").alias("precSum")) Result = Result
Result = Result.orderBy(["year", "month"], ascending=[0,0])
Result.saveAsTextFile("result5")
year month prec
2016 05 29.250000000000007
2016 02 21.5625
2015 12 28.925
2015 11 63.88750000000002
```

```
2015 09 101.29999999999998
2015 08 26.9875
2015 07 119.09999999999995
2015 06 78.66250000000001
2015 05 93.225
2015 04 15.3375
...
2013 07 54.56249999999999
2013 05 47.92500000000001
2013 04 38.287500000000016
2013 02 25.52500000000002
2012 11 68.65
2012 10 65.58333333333334
2012 09 72.75
2012 07 59.06666666666667
2012 04 62.78333333333335
2012 01 43.55000000000003
```