732A54: Lab 2
Big Data Analytics

Carles Sans Fuentes

March 14, 2017

Assignment 1

Exercise 1

Part A code

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

sc = SparkContext(appName = "max temperatures descending SQL") # importing mySpark context from pyspark
sqlContext = SQLContext(sc) # running sql from my sar context

rdd= sc.textFile("/user/x_carsa/data/temperature-readings.csv")
parts = rdd.map(lambda l: l.split(";"))
tempReadings= parts.map(lambda p: Row(station=p[0],\
                                      date=p[1], year=p[1].split("-")[0],\
                                       time=p[2], value=float(p[3]), quality=p[4]))

schemaTempReadings= sqlContext.createDataFrame(tempReadings)
schemaTempReadings.registerTempTable("TempReadings")

maxtempdesc = sqlContext.sql("SELECT distinct(one.year) as year, first(one.station) as station, first(one.value) as value \
                        FROM TempReadings as one \
                          INNER JOIN \
                            (SELECT year, MAX(value) AS maxvalue \
                            FROM TempReadings \
                            WHERE year between 1950 and 2014\
                            GROUP BY year) as two \
                          ON one.year = two.year \
                          WHERE one.value = two.maxvalue \
                        GROUP BY one.year \
                        ORDER BY value DESC")
maxtempdesc = maxtempdesc.rdd.repartition(1)\
                            .sortBy(ascending = False, keyfunc = lambda \
                            (year, station, value): value)
print maxtempdesc.take(20)


maxtempdesc.saveAsTextFile("results/1_resultsMaxSQL")
```

Output:

```

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

```
Row(year=u'1986', station=u'76470', value=33.2)
Row(year=u'1956', station=u'145340', value=33.0)
Row(year=u'2000', station=u'62400', value=33.0)
Row(year=u'1959', station=u'65160', value=32.8)
Row(year=u'1991', station=u'137040', value=32.7)
Row(year=u'2006', station=u'75240', value=32.7)
Row(year=u'1988', station=u'102540', value=32.6)
Row(year=u'2011', station=u'172770', value=32.5)
Row(year=u'1999', station=u'98210', value=32.4)
Row(year=u'1955', station=u'97260', value=32.2)
Row(year=u'2008', station=u'82090', value=32.2)
Row(year=u'1973', station=u'71470', value=32.2)
```

Part b code

```

#min
mintempdesc = sqlContext.sql("SELECT distinct(one.year) as year, first(one.station) as station,
      first(one.value) as value \
                        FROM TempReadings as one \
                          INNER JOIN \
                            (SELECT year, MIN(value) AS maxvalue \
                            FROM TempReadings \
                            WHERE year between 1950 and 2014\
                            GROUP BY year) as two \
                          ON one.year = two.year \
                          WHERE one.value = two.maxvalue \
                        GROUP BY one.year \
                        ORDER BY value DESC")
mintempdesc = mintempdesc.rdd.repartition(1)\
                            .sortBy(ascending = False, keyfunc = lambda \
                            (year, station, value): value)
print mintempdesc.take(20)
mintempdesc.saveAsTextFile("results/1_resultsMinSQL")
```

Output:

```

Row(year=u'1990', station=u'147270', value=-35.0)
Row(year=u'1952', station=u'192830', value=-35.5)
Row(year=u'1974', station=u'179950', value=-35.6)
Row(year=u'1954', station=u'113410', value=-36.0)
Row(year=u'1992', station=u'179960', value=-36.1)
Row(year=u'1975', station=u'157860', value=-37.0)
Row(year=u'1972', station=u'167860', value=-37.5)
Row(year=u'2000', station=u'169860', value=-37.6)
Row(year=u'1995', station=u'182910', value=-37.6)
Row(year=u'1957', station=u'159970', value=-37.8)
Row(year=u'1983', station=u'191900', value=-38.2)
Row(year=u'1989', station=u'166870', value=-38.2)
Row(year=u'1953', station=u'183760', value=-38.4)
Row(year=u'2009', station=u'179960', value=-38.5)
Row(year=u'1993', station=u'191900', value=-39.0)
Row(year=u'1984', station=u'123480', value=-39.2)
Row(year=u'1973', station=u'166870', value=-39.3)
Row(year=u'2008', station=u'179960', value=-39.3)
Row(year=u'1991', station=u'179960', value=-39.3)
Row(year=u'2005', station=u'155790', value=-39.4)
Row(year=u'1961', station=u'181900', value=-39.5)
Row(year=u'1964', station=u'166810', value=-39.5)
Row(year=u'1970', station=u'179950', value=-39.6)
Row(year=u'2004', station=u'166940', value=-39.7)
Row(year=u'1988', station=u'170790', value=-39.9)
Row(year=u'1960', station=u'155910', value=-40.0)
Row(year=u'1997', station=u'179960', value=-40.2)
Row(year=u'1994', station=u'179960', value=-40.5)
Row(year=u'2006', station=u'169860', value=-40.6)
Row(year=u'2007', station=u'169860', value=-40.7)
Row(year=u'2013', station=u'179960', value=-40.7)
Row(year=u'1963', station=u'181900', value=-41.0)
Row(year=u'1955', station=u'160790', value=-41.2)
Row(year=u'1969', station=u'181900', value=-41.5)
Row(year=u'2003', station=u'179960', value=-41.5)
Row(year=u'2010', station=u'191910', value=-41.7)
Row(year=u'1996', station=u'155790', value=-41.7)
Row(year=u'2011', station=u'179960', value=-42.0)
Row(year=u'1962', station=u'181900', value=-42.0)
Row(year=u'1950', station=u'155910', value=-42.0)
Row(year=u'1968', station=u'179950', value=-42.0)
Row(year=u'1951', station=u'155910', value=-42.0)
Row(year=u'1976', station=u'192830', value=-42.2)
Row(year=u'1982', station=u'113410', value=-42.2)
```

```
Row(year=u'2002', station=u'169860', value=-42.2)
Row(year=u'1977', station=u'179950', value=-42.5)
Row(year=u'2014', station=u'192840', value=-42.5)
Row(year=u'1998', station=u'169860', value=-42.7)
Row(year=u'2012', station=u'191910', value=-42.7)
Row(year=u'1958', station=u'159970', value=-43.0)
Row(year=u'1985', station=u'159970', value=-43.4)
Row(year=u'1959', station=u'159970', value=-43.6)
Row(year=u'1965', station=u'189780', value=-44.0)
Row(year=u'1981', station=u'166870', value=-44.0)
Row(year=u'2001', station=u'112530', value=-44.0)
Row(year=u'1979', station=u'112170', value=-44.0)
Row(year=u'1986', station=u'167860', value=-44.2)
Row(year=u'1971', station=u'166870', value=-44.3)
Row(year=u'1980', station=u'191900', value=-45.0)
Row(year=u'1956', station=u'160790', value=-45.0)
Row(year=u'1967', station=u'166870', value=-45.4)
Row(year=u'1987', station=u'123480', value=-47.3)
Row(year=u'1978', station=u'155940', value=-47.7)
Row(year=u'1999', station=u'192830', value=-49.0)
Row(year=u'1966', station=u'179950', value=-49.4)
```

Part A and B in API

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

sc = SparkContext(appName = "max temperatures descending SQL") # importing mySpark context from
     pyspark
sqlContext = SQLContext(sc) # running sql from my sar context

file= sc.textFile("/user/x_carsa/data/temperature-readings.csv")
parts = file.map(lambda l: l.split(";"))
tempReadings= parts.map(lambda p: Row(station=p[0],\
                                      date=p[1], year=p[1].split("-")[0],\
                                       time=p[2], value=float(p[3]), quality=p[4]))

schemaTempReadings= sqlContext.createDataFrame(tempReadings)
schemaTempReadings.registerTempTable("TempReadings")

maxtempdesc = schemaTempReadings\
    .filter(schemaTempReadings.year>=1950)\
    .filter(schemaTempReadings.year<=2014)\
    .groupBy(schemaTempReadings.year,schemaTempReadings.station)\
    .agg(F.max(schemaTempReadings.value).alias("maxtemp"))\
    .rdd.repartition(1)\
    .sortBy(ascending = False, keyfunc = lambda a: a[1],  numPartitions=1)



mintempdesc = schemaTempReadings\
    .filter(schemaTempReadings.year>=1950)\
    .filter(schemaTempReadings.year<=2014)\
    .groupBy(schemaTempReadings.year,schemaTempReadings.station)\
    .agg(F.min(schemaTempReadings.value).alias("mintemp"))\
    .rdd.repartition(1)\
    .sortBy(ascending = False, keyfunc = lambda a: a[1],  numPartitions=1)

maxtempdesc.saveAsTextFile("results/1_resultsMaxAPI")
mintempdesc.saveAsTextFile("results/1_resultsMinAPI")
```

Exercise 2

Count the number of readings for each month in the period of 1950-2014 which are higher than 10 degrees.

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

sc = SparkContext(appName = "counts over 10 degreesSQL") # importing mySpark context from
    pyspark
sqlContext = SQLContext(sc) # running sql from my sar context
rdd= sc.textFile("/user/x_carsa/data/temperature-readings.csv")

parts = rdd.map(lambda l: l.split(";"))
tempReadings= parts.map(lambda p: Row(station=p[0],\
```

```
                                    month=p[1].split("-")[1], year=p[1].split("-")[0],\
                                     time=p[2], value=float(p[3]), quality=p[4]))

schemaTempReadings= sqlContext.createDataFrame(tempReadings)
schemaTempReadings.registerTempTable("tempReadings")

largerThan10Degrees = sqlContext.sql("SELECT year, month, count(value) as value\
                     FROM tempReadings\
                     WHERE year between 1950 and 2014 and value>=10.0\
                     group by year, month")
largerThan10Degrees = largerThan10Degrees.rdd.repartition(1)\
                            .sortBy(ascending = False, keyfunc = lambda \
                            (year, month, value): value)
print largerThan10Degrees.take(20)
largerThan10Degrees.saveAsTextFile("results/2_largerthan10degreesSQL")
```

Output:

```
(u'2014-07', 147681)
(u'2011-07', 146656)
(u'2010-07', 143419)
(u'2012-07', 137477)
(u'2013-07', 133657)
(u'2009-07', 133008)
(u'2011-08', 132734)
(u'2009-08', 128349)
(u'2013-08', 128235)
(u'2003-07', 128133)
(u'2002-07', 127956)
(u'2006-08', 127622)
(u'2008-07', 126973)
(u'2002-08', 126073)
(u'2005-07', 125294)
(u'2011-06', 125193)
(u'2012-08', 125037)
(u'2006-07', 124794)
(u'2010-08', 124417)
(u'2014-08', 124045)
(u'1997-07', 123496)
(u'2007-07', 123218)
(u'2013-06', 122181)
(u'1997-08', 121154)
(u'2001-07', 120529)
(u'1998-07', 120230)
(u'2000-07', 119769)
(u'2004-07', 119536)
(u'1999-07', 116385)
(u'2008-08', 114272)
(u'2004-08', 114168)
(u'2002-06', 114034)
(u'2005-08', 113950)
(u'2001-08', 113937)
(u'2007-08', 110428)
(u'2000-08', 109201)
(u'2003-08', 108501)
(u'1996-08', 107758)
(u'1997-06', 104696)
(u'1999-06', 103227)
(u'2007-06', 103046)
(u'2008-06', 102900)
(u'2010-06', 102716)
(u'2006-06', 102588)
(u'2014-06', 101711)
(u'1998-08', 101387)
(u'1996-07', 99916)
(u'2003-06', 99693)
(u'2011-09', 99335)
(u'1999-08', 97437)
(u'2006-09', 97181)
(u'2012-06', 94513)
(u'2001-06', 93375)
(u'2005-06', 90724)
(u'2004-06', 89628)
(u'1999-09', 89418)
(u'2009-09', 89106)
(u'2009-06', 87787)
(u'2000-06', 86592)
(u'2014-09', 86090)
(u'1998-06', 82608)
(u'2013-05', 81996)
(u'2013-09', 81960)
(u'1996-06', 80440)
(u'2001-09', 79657)
(u'1998-09', 76535)
(u'1988-07', 75521)
```

```
(u'2005-09', 75494)
(u'2010-09', 74816)
(u'1997-09', 74472)
(u'1991-07', 73385)
(u'2004-09', 73334)
(u'1973-07', 71522)
(u'1991-08', 71185)
(u'2003-09', 70459)
(u'2012-09', 70427)
(u'1990-07', 70031)
(u'1988-08', 69913)
(u'1987-07', 68135)
(u'1989-07', 67880)
(u'1989-08', 67793)
(u'1990-08', 67604)
(u'1995-08', 66920)
(u'1974-07', 66277)
(u'2002-05', 66116)
(u'2002-09', 65928)
(u'1974-08', 64470)
(u'1975-07', 64408)
(u'1976-07', 64109)
(u'2000-09', 63837)
(u'1988-06', 63572)
(u'1992-07', 62911)
(u'1975-08', 62565)
(u'2007-09', 61346)
(u'1978-07', 60998)
(u'2008-09', 60989)
(u'1976-08', 60898)
(u'2009-05', 60867)
(u'1989-06', 60822)
(u'1979-07', 60719)
(u'1994-07', 60691)
```

Part b: 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
from pyspark.sql import SQLContext, Row
from pyspark.sql import functions as F


sc = SparkContext(appName = "counts over 10 degreesSQL one for each station") # importing
    mySpark context from pyspark
sqlContext = SQLContext(sc) # running sql from my sar context
rdd= sc.textFile("/user/x_carsa/data/temperature-readings.csv")

parts = rdd.map(lambda l: l.split(";"))
tempReadings= parts.map(lambda p: Row(station=p[0],\
                                      month=p[1].split("-")[1], year=p[1].split("-")[0],\
                                       time=p[2], value=float(p[3]), quality=p[4]))

schemaTempReadings= sqlContext.createDataFrame(tempReadings)
schemaTempReadings.registerTempTable("tempReadings")

largerThan10Degrees = sqlContext.sql("SELECT FIRST(year), FIRST(month), FIRST(station), COUNT(
    DISTINCT value) as counts\
                    FROM tempReadings\
                    WHERE year between 1950 and 2014 and value>=10.0\
                    group by year, month, station\
                    ORDER BY counts DESC")

largerThan10Degrees = largerThan10Degrees.rdd.repartition(1)\
                            .map(lambda (year, month, station, counts): \
                                ((year, month), counts)) \
                            .reduceByKey(lambda  \
                                    count1, count2: count1 + count2) \
                            .sortBy(ascending = False, keyfunc = lambda \
                                    ((year, month), counts): counts)

print largerThan10Degrees.take(20)

largerThan10Degrees.saveAsTextFile("results/2_largerthan10degreesSQLoneforeachstationSQL")
```

Output:

```
(u'1972-10', 378)
(u'1973-06', 377)
(u'1973-05', 377)
```

```
 4 (u'1973-09', 376)
 5 (u'1972-08', 376)
 6 (u'1972-05', 375)
 7 (u'1971-08', 375)
 8 (u'1972-06', 375)
 9 (u'1972-09', 375)
10 (u'1971-09', 374)
11 (u'1972-07', 374)
12 (u'1971-06', 374)
13 (u'1973-08', 373)
14 (u'1971-05', 373)
15 (u'1974-06', 372)
16 (u'1974-08', 372)
17 (u'1974-05', 370)
18 (u'1970-08', 370)
19 (u'1971-07', 370)
20 (u'1973-07', 370)
21 (u'1974-09', 370)
22 (u'1975-09', 369)
23 (u'1970-09', 369)
24 (u'1976-05', 369)
25 (u'1970-06', 369)
26 (u'1976-06', 368)
27 (u'1975-06', 368)
28 (u'1975-08', 367)
29 (u'1975-05', 367)
30 (u'1970-05', 366)
31 (u'1976-09', 365)
32 (u'1977-06', 364)
33 (u'1967-05', 363)
34 (u'1976-08', 363)
35 (u'1974-07', 362)
36 (u'1970-07', 362)
37 (u'1967-09', 361)
38 (u'1966-09', 360)
39 (u'1966-06', 360)
40 (u'1966-08', 359)
41 (u'1969-09', 359)
42 (u'1967-06', 359)
43 (u'1965-09', 358)
44 (u'1978-09', 358)
45 (u'1967-08', 358)
46 (u'1975-07', 358)
47 (u'1969-08', 357)
48 (u'1968-06', 357)
49 (u'1968-08', 357)
50 (u'1976-07', 356)
51 (u'1968-09', 356)
52 (u'1968-05', 355)
53 (u'1965-06', 355)
54 (u'1979-05', 354)
55 (u'1978-06', 354)
56 (u'1965-08', 354)
57 (u'1966-05', 354)
58 (u'1977-08', 354)
59 (u'1968-07', 353)
60 (u'1977-09', 353)
61 (u'1978-05', 352)
62 (u'1969-06', 352)
63 (u'1966-07', 352)
64 (u'1967-07', 351)
65 (u'1979-06', 351)
66 (u'1977-05', 351)
67 (u'1979-09', 351)
68 (u'1977-07', 350)
69 (u'1978-08', 350)
70 (u'1965-07', 349)
71 (u'1973-10', 349)
72 (u'1969-07', 349)
73 (u'1971-10', 347)
74 (u'1969-10', 346)
75 (u'1979-07', 345)
76 (u'1996-06', 345)
77 (u'1970-10', 345)
78 (u'1974-04', 344)
79 (u'1965-05', 344)
80 (u'1978-07', 343)
81 (u'1996-07', 342)
82 (u'1996-05', 342)
83 (u'1996-08', 341)
84 (u'1978-10', 340)
85 (u'1996-09', 340)
86 (u'1975-10', 340)
87 (u'1979-08', 340)
88 (u'1997-09', 340)
89 (u'1982-06', 339)
90 (u'1997-06', 338)
```

```
 91 (u'1980-09', 338)
 92 (u'1980-05', 337)
 93 (u'1981-05', 337)
 94 (u'1997-08', 337)
 95 (u'1983-06', 337)
 96 (u'1983-05', 336)
 97 (u'1965-10', 335)
 98 (u'1981-09', 335)
 99 (u'1969-05', 335)
100 (u'1981-08', 334)
101 (u'1982-09', 334)
102 (u'1997-07', 333)
103 (u'1984-05', 333)
104 (u'1983-09', 332)
105 (u'1980-06', 332)
106 (u'1981-06', 331)
107 (u'1999-06', 330)
108 (u'1983-08', 330)
109 (u'1982-05', 330)
110 (u'1980-08', 330)
111 (u'1999-07', 329)
112 (u'1981-07', 329)
113 (u'1999-09', 328)
114 (u'1985-09', 327)
115 (u'1984-09', 327)
116 (u'1999-08', 327)
117 (u'1998-09', 326)
118 (u'1998-08', 326)
119 (u'2002-06', 326)
120 (u'1998-07', 326)
121 (u'1982-08', 326)
122 (u'1998-06', 326)
123 (u'1981-10', 325)
124 (u'1999-05', 325)
125 (u'2000-08', 325)
126 (u'1985-05', 325)
127 (u'1980-07', 324)
128 (u'1967-10', 324)
129 (u'1984-06', 324)
130 (u'2001-07', 324)
131 (u'2002-07', 324)
132 (u'2001-06', 324)
133 (u'1985-06', 324)
134 (u'2002-05', 324)
135 (u'1987-06', 323)
136 (u'2003-06', 323)
137 (u'2000-05', 323)
138 (u'2002-09', 323)
139 (u'2001-08', 323)
140 (u'1986-09', 323)
141 (u'1987-09', 323)
142 (u'2002-08', 322)
143 (u'2001-09', 322)
144 (u'1968-04', 322)
145 (u'1998-05', 322)
146 (u'2000-09', 322)
147 (u'1988-06', 322)
148 (u'2003-05', 321)
149 (u'2004-05', 321)
150 (u'2003-07', 321)
151 (u'1984-10', 321)
152 (u'1982-07', 321)
153 (u'2000-06', 321)
154 (u'1991-06', 321)
155 (u'2004-09', 321)
156 (u'1987-05', 320)
157 (u'2010-06', 320)
158 (u'2000-07', 320)
159 (u'1988-05', 320)
160 (u'2003-09', 320)
161 (u'2004-08', 320)
162 (u'1987-08', 320)
163 (u'2003-08', 320)
164 (u'1997-05', 319)
165 (u'1987-07', 319)
166 (u'2004-06', 319)
167 (u'2004-07', 319)
168 (u'2010-05', 319)
169 (u'2011-07', 319)
170 (u'1983-07', 319)
171 (u'2010-07', 318)
```

Exercise 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.

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


sc = SparkContext(appName = "average temperature for each month SQL") # importing mySpark
     context from pyspark
sqlContext = SQLContext(sc) # running sql from my sar context
rdd= sc.textFile("/user/x_carsa/data/temperature-readings.csv")

parts = rdd.map(lambda l: l.split(";"))
tempReadings= parts.map(lambda p: Row(station=p[0],\
                                      month=p[1].split("-")[1], year=p[1].split("-")[0],\
                                       time=p[2], value=float(p[3]), quality=p[4]))

schemaTempReadings= sqlContext.createDataFrame(tempReadings)
schemaTempReadings.registerTempTable("tempReadings")

averagetemperature = sqlContext.sql("SELECT year, month, station, avg(value) as avgvalue \
                    FROM tempReadings \
                    WHERE year between 1960 and 2014 \
                    group by year, month, station \
                    order by avgvalue DESC")

averagetemperature = averagetemperature.rdd.repartition(1)\
                            .sortBy(ascending = False, keyfunc = lambda \
                            (year, month, station,  avgvalue): avgvalue)

averagetemperature.saveAsTextFile("results/3_average_temperature_eachmonthSQL")
```

Output:

```
((u'2014-07', u'96000'), 26.3)
((u'1994-07', u'96550'), 23.071052631578944)
((u'1983-08', u'54550'), 23.0)
((u'1994-07', u'78140'), 22.970967741935482)
((u'1994-07', u'85280'), 22.872580645161293)
((u'1994-07', u'75120'), 22.85806451612903)
((u'1994-07', u'65450'), 22.856451612903225)
((u'1994-07', u'96000'), 22.80806451612903)
((u'1994-07', u'95160'), 22.76451612903226)
((u'1994-07', u'86200'), 22.711290322580645)
((u'2002-08', u'78140'), 22.700000000000003)
((u'1994-07', u'76000'), 22.698387096774198)
((u'1997-08', u'78140'), 22.666129032258066)
((u'1994-07', u'105260'), 22.65967741935484)
((u'1975-08', u'54550'), 22.642857142857142)
((u'2006-07', u'76530'), 22.598387096774193)
((u'1994-07', u'86330'), 22.54838709677419)
((u'2006-07', u'75120'), 22.52741935483871)
((u'1994-07', u'54300'), 22.469354838709677)
((u'2006-07', u'78140'), 22.45806451612903)
((u'2001-07', u'96550'), 22.408333333333335)
((u'2010-07', u'98180'), 22.37903225806452)
((u'2006-07', u'65450'), 22.37741935483871)
((u'1994-07', u'85210'), 22.375806451612902)
((u'1994-07', u'98180'), 22.367741935483874)
((u'2014-07', u'98180'), 22.367741935483874)
((u'2002-08', u'98180'), 22.366129032258062)
((u'1994-07', u'92100'), 22.31774193548387)
((u'1994-07', u'86470'), 22.30806451612903)
((u'1994-07', u'83230'), 22.272580645161288)
((u'1994-07', u'64290'), 22.259677419354837)
((u'1994-07', u'97490'), 22.258064516129032)
((u'1994-07', u'94180'), 22.25322580645161)
((u'1972-07', u'173960'), 22.244999999999997)
((u'1994-07', u'74080'), 22.241935483870968)
((u'2006-07', u'54300'), 22.23709677419355)
((u'2002-08', u'98210'), 22.23548387096774)
((u'1994-07', u'106070'), 22.232258064516135)
((u'1994-07', u'75100'), 22.229032258064517)
((u'1994-07', u'53440'), 22.197499999999998)
((u'1994-07', u'83270'), 22.177419354838705)
((u'1994-07', u'103080'), 22.16451612903226)
((u'1994-07', u'82110'), 22.161290322580644)
```

```
((u'1994-07', u'97120'), 22.135483870967743)
((u'2010-07', u'98210'), 22.111290322580647)
((u'1994-07', u'53430'), 22.096774193548388)
((u'1997-08', u'86330'), 22.07903225806452)
((u'2006-07', u'66500'), 22.05483870967742)
((u'1994-07', u'76530'), 22.033870967741933)
((u'1997-08', u'98210'), 21.983870967741936)
((u'2014-07', u'98210'), 21.962903225806453)
((u'1994-07', u'62400'), 21.951612903225808)
((u'1997-08', u'62400'), 21.938709677419357)
((u'1994-07', u'108110'), 21.90806451612903)
((u'1994-07', u'83130'), 21.900000000000002)
((u'1997-08', u'98180'), 21.887096774193544)
((u'2006-07', u'98210'), 21.872580645161293)
((u'1997-08', u'98290'), 21.864516129032257)
((u'1991-08', u'78040'), 21.85)
((u'2010-07', u'78140'), 21.830645161290324)
((u'1994-07', u'63340'), 21.7758064516129)
((u'1994-07', u'91130'), 21.76290322580645)
((u'1994-07', u'105370'), 21.761290322580642)
((u'2008-07', u'83420'), 21.75)
((u'1994-07', u'64130'), 21.72741935483871)
((u'1997-08', u'96550'), 21.725806451612904)
((u'1994-07', u'83440'), 21.716129032258067)
((u'2006-07', u'85210'), 21.706451612903226)
((u'1994-07', u'74420'), 21.690322580645162)
((u'2003-07', u'98180'), 21.68548387096774)
((u'1997-08', u'52230'), 21.68548387096774)
```

Exercise 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 200 mm. In this exercise you will use the temperature-readings.csv and precipitation-readings.csv file.

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

sc = SparkContext(appName = "counts over 10 degreesSQL one for each station") # importing
    mySpark context from pyspark
sqlContext = SQLContext(sc) # running sql from my sar context

##Reading temperatures document
rdd= sc.textFile("/user/x_carsa/data/temperature-readings.csv")
parts = rdd.map(lambda l: l.split(";"))
tempReadings= parts.map(lambda obs: Row(station = obs[0], \
                    date = obs[1],  \
                    year = obs[1].split("-")[0], \
                    month = obs[1].split("-")[1], \
                    day = obs[1].split("-")[2], \
                    time = obs[2], \
                    temp = float(obs[3]), \
                    quality = obs[4]))
schemaTempReadings= sqlContext.createDataFrame(tempReadings)
schemaTempReadings.registerTempTable("tempReadings")

##Reading precipitations document
rdd2= sc.textFile("/user/x_carsa/data/precipitation-readings.csv")
parts2 = rdd2.map(lambda l: l.split(";"))
precReadings= parts2.map(lambda obs: Row(station = obs[0], \
                    date = obs[1],  \
                    year = obs[1].split("-")[0], \
                    month = obs[1].split("-")[1], \
                    day = obs[1].split("-")[2], \
                    time = obs[2], \
                    prec = float(obs[3]), \
                    quality = obs[4]))
schemaTempReadings= sqlContext.createDataFrame(precReadings)
schemaTempReadings.registerTempTable("precReadings")

averagetemperature = sqlContext.sql("SELECT tempReadings.station,  \
                    MAX(tempReadings.temp) as maxtemp, MAX(dailyPrec) AS maxprec \
                   FROM tempReadings, \
                        (SELECT station, date, SUM(prec) as dailyPrec \
                        FROM precReadings \
                        WHERE precReadings.year between 1960 and 2014 \
                        GROUP BY station, date ) prectable \
                    WHERE   tempReadings.station = prectable.station AND\
```

```
          tempReadings.date = prectable.date AND\
          tempReadings.year between 1960 and 2014 AND \
                            temp between 25.0 and 30.0 AND\
                            dailyPrec between 100 and 200 \
             GROUP BY tempReadings.station \
                   ORDER BY tempReadings.station DESC")


averagetemperature = averagetemperature.rdd.repartition(1) \
                     .sortBy(ascending = False, keyfunc = lambda \
                         (station, temp, prec): station)

averagetemperature.saveAsTextFile("results/4_stations_with_max_temperatures_precipitationsSQL")
print averagetemperature.take(20)
```

Output:

```
No output
```

Exercise 5

Calculate the average monthly precipitation for the Östergotland region (list of stations is provided in the separate file). In order to do this, you will first need to calculate the total daily precipitation before calculating the monthly average. In this exercise you will use the precipitation-readings.csv and stations-Ostergotland.csv files.

```
##5
##5

from pyspark import SparkContext
from pyspark.sql import SQLContext, Row
from pyspark.sql import functions as F

sc = SparkContext(appName = "counts over 10 degreesSQL one for each station") # importing
     mySpark context from pyspark
sqlContext = SQLContext(sc) # running sql from my sar context


reading2 = sc.textFile("/user/x_carsa/data/stations-Ostergotland.csv")
separate2 = reading2.map(lambda a: a.split(";"))
ostgota_stations =  separate2.map(lambda obs: Row(station=obs[0],\
        name=obs[1],\
        height=float(obs[2]),\
        lat=float(obs[3]),\
        lon=float(obs[4]),\
        date_from=obs[5],\
        date_to=obs[6],\
        elevation=float(obs[7])))
stationReadings= sqlContext.createDataFrame(ostgota_stations)
stationReadings.registerTempTable("stationReadings")


##Reading precipitations document
rdd2= sc.textFile("/user/x_carsa/data/precipitation-readings.csv")
parts2 = rdd2.map(lambda l: l.split(";"))
precReadings= parts2.map(lambda obs: Row(station = obs[0], \
                    date = obs[1],  \
                    year = obs[1].split("-")[0], \
                    month = obs[1].split("-")[1], \
        yearmonth = obs[1][:7], \
                    day = obs[1].split("-")[2], \
                    time = obs[2], \
                    prec = float(obs[3]), \
                    quality = obs[4]))
schemaTempReadings= sqlContext.createDataFrame(precReadings)
schemaTempReadings.registerTempTable("precReadings")




averagetemperature = sqlContext.sql("SELECT prectable.year, prectable.month,\
                     avg(dailyPrec) AS avgprec \
                    FROM  \
                         (SELECT year, month, stationReadings.station , sum(prec) as dailyPrec \
                         FROM precReadings, stationReadings \
      WHERE precReadings.station = stationReadings.station \
                         GROUP BY year, month, stationReadings.station) AS prectable \
```

```
                    WHERE   prectable.year between 1993 and 2016  \
                    GROUP BY prectable.year, prectable.month")


averagetemperature = averagetemperature.rdd.repartition(1) \
                    .sortBy(ascending = False, keyfunc = lambda \
                        (year, month, prec): (year,month))
print averagetemperature.take(20)
averagetemperature.saveAsTextFile("results/5_average_montly_prec_SQL")
```

Output:

```
[Row(year=u'2016', month=u'07', avgprec=0.0), Row(year=u'2016', month=u'06', avgprec=47.6625),
    Row(year=u'2016', month=u'05', avgprec=29.250000000000004), Row(year=u'2016', month=u'04',
    avgprec=26.900000000000006), Row(year=u'2016', month=u'03', avgprec=19.962500000000002),
    Row(year=u'2016', month=u'02', avgprec=21.5625), Row(year=u'2016', month=u'01', avgprec
    =22.325), Row(year=u'2015', month=u'12', avgprec=28.925000000000004), Row(year=u'2015',
    month=u'11', avgprec=63.88750000000002), Row(year=u'2015', month=u'10', avgprec=2.2625),
    Row(year=u'2015', month=u'09', avgprec=101.29999999999998), Row(year=u'2015', month=u'08',
    avgprec=26.987499999999997), Row(year=u'2015', month=u'07', avgprec=119.09999999999997),
    Row(year=u'2015', month=u'06', avgprec=78.66250000000001), Row(year=u'2015', month=u'05',
    avgprec=93.22499999999998), Row(year=u'2015', month=u'04', avgprec=15.337499999999999), Row
    (year=u'2015', month=u'03', avgprec=42.612500000000004), Row(year=u'2015', month=u'02',
    avgprec=24.825), Row(year=u'2015', month=u'01', avgprec=59.11250000000003), Row(year=u'2014
    ', month=u'12', avgprec=35.46250000000001)]
```

Exercise 6

Compare the average monthly temperature (find the difference) in the period 1950-2014 for each station in Östergotland with long-term monthly averages in the period of 1950-1980. Make a plot of your results.

```
##6

from pyspark import SparkContext
from pyspark.sql import SQLContext, Row
from pyspark.sql import functions as F

sc = SparkContext(appName = "counts over 10 degreesSQL one for each station") # importing
    mySpark context from pyspark
sqlContext = SQLContext(sc) # running sql from my sar context
reading2 = sc.textFile("/user/x_carsa/data/stations-Ostergotland.csv")
separate2 = reading2.map(lambda a: a.split(";"))
stations = separate2.map(lambda observation: int(observation[0]))
stations = stations.distinct().collect() #collect transforms the rdd to a python list object
stations = {station: True for station in stations}

##Reading temperatures document
rdd= sc.textFile("/user/x_carsa/data/temperature-readings.csv")
parts = rdd.map(lambda l: l.split(";"))
tempReadings= parts.filter(lambda obs: stations.get(int(obs[0]), False)) \
                    .map(lambda obs: Row(station = obs[0], \
                    date = obs[1],  \
                    year = obs[1].split("-")[0], \
                    month = obs[1].split("-")[1], \
                    yearmonth = obs[1][:7], \
                    day = obs[1].split("-")[2], \
                    time = obs[2], \
                    temp = float(obs[3]), \
                    quality = obs[4]))
schemaTempReadings= sqlContext.createDataFrame(tempReadings)
schemaTempReadings.registerTempTable("tempReadings")


shorttemperature = sqlContext.sql("SELECT yearmonth,\
                    avg(double_dailytemp)/2 as avgtemp\
                   FROM (SELECT yearmonth, year, month, date, min(temp) + max(temp) AS double_
                       dailytemp \
                     FROM tempReadings \
                     WHERE year between 1950 and 2014 \
                     GROUP BY yearmonth, year, date, station, month ) as tempdaily \
                   GROUP BY yearmonth")

longtemperature = shorttemperature.filter(F.substring(shorttemperature["yearmonth"], 1, 4) <=
    1980) \
                    .groupBy(F.substring(shorttemperature["yearmonth"], 6, 7).alias("monthpart"
                        )) \
```

```
                    .agg(F.avg(shorttemperature["avgtemp"]).alias("avglongtemp"))


jointables = shorttemperature.join(longtemperature, (F.substring(shorttemperature[0], 6, 7) ==
    longtemperature["monthpart"]), "inner")
print jointables.take(20)
av_diff = jointables.select(jointables["yearmonth"],(F.abs(jointables["avgtemp"]) - F.abs(
    jointables["avglongtemp"])))

averagetemperature = av_diff.rdd.repartition(1) \
                    .sortBy(ascending = False, keyfunc = lambda \
                        (yearmonth, tempdiff): (yearmonth))

print averagetemperature.take(20)
averagetemperature.saveAsTextFile("results/6_average_montly_diff_SQL")
```

Output: A graphic and a take of the results is shown below:

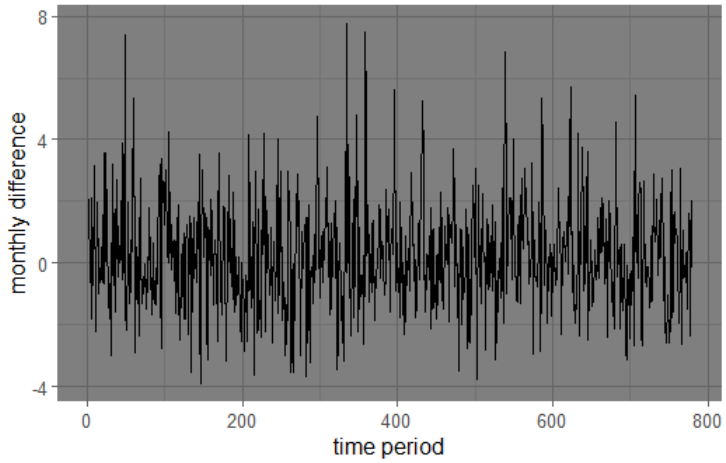

Figure 1: Average monthly difference on temperature

```
[Row(yearmonth=u'2014-12', (abs(avgtemp) - abs(avglongtemp))=-0.7938517834097852), Row(
    yearmonth=u'2014-11', (abs(avgtemp) - abs(avglongtemp))=2.063539672692899), Row(yearmonth=u
    '2014-10', (abs(avgtemp) - abs(avglongtemp))=1.521957490617968), Row(yearmonth=u'2014-09',
    (abs(avgtemp) - abs(avglongtemp))=0.06105818643722927), Row(yearmonth=u'2014-08', (abs(
    avgtemp) - abs(avglongtemp))=-0.6426470719706874), Row(yearmonth=u'2014-07', (abs(avgtemp)
    - abs(avglongtemp))=2.105921838713968), Row(yearmonth=u'2014-06', (abs(avgtemp) - abs(
    avglongtemp))=-1.8073686197315322), Row(yearmonth=u'2014-05', (abs(avgtemp) - abs(
    avglongtemp))=0.267190650140698), Row(yearmonth=u'2014-04', (abs(avgtemp) - abs(avglongtemp
    ))=2.0661931589915428), Row(yearmonth=u'2014-03', (abs(avgtemp) - abs(avglongtemp))
    =3.176498950234641), Row(yearmonth=u'2014-02', (abs(avgtemp) - abs(avglongtemp))
    =-2.2292398859946143), Row(yearmonth=u'2014-01', (abs(avgtemp) - abs(avglongtemp))
    =-0.9325880207201753), Row(yearmonth=u'2013-12', (abs(avgtemp) - abs(avglongtemp))
    =1.9232603493728844), Row(yearmonth=u'2013-11', (abs(avgtemp) - abs(avglongtemp))
    =0.9342517939050197), Row(yearmonth=u'2013-10', (abs(avgtemp) - abs(avglongtemp))
    =0.752309396776325), Row(yearmonth=u'2013-09', (abs(avgtemp) - abs(avglongtemp))
    =-0.9757232929529476), Row(yearmonth=u'2013-08', (abs(avgtemp) - abs(avglongtemp))
    =-0.3146412068680515), Row(yearmonth=u'2013-07', (abs(avgtemp) - abs(avglongtemp))
    =0.02029134018023626), Row(yearmonth=u'2013-06', (abs(avgtemp) - abs(avglongtemp))
    =-0.54418680154971), Row(yearmonth=u'2013-05', (abs(avgtemp) - abs(avglongtemp))
    =1.5739208554192992)]
```