TDDE31 Lab2

Kristian Sikiric (krisi211) & Pontus Svensson (ponsv690)

1)

Highest:

```
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)
…
```

Lowest:

```
Row(year=u'1990', station=u'166870', value=-35.0)
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'1974', station=u'166870', value=-35.6)
Row(year=u'1954', station=u'113410', value=-36.0)
…
```

2)

a)

```
Row(year=u'2014', month=u'07', value=147910)
Row(year=u'2011', month=u'07', value=147060)
Row(year=u'2010', month=u'07', value=143860)
Row(year=u'2012', month=u'07', value=138166)
…
Row(year=u'1964', month=u'02', value=1)
Row(year=u'1958', month=u'01', value=1)
Row(year=u'1970', month=u'03', value=1)
Row(year=u'1971', month=u'02', value=1)
```

b)

```
Row(year=u'1972', month=u'10', count=378)
Row(year=u'1973', month=u'05', count=377)
Row(year=u'1973', month=u'06', count=377)
Row(year=u'1973', month=u'09', count=376)
Row(year=u'1972', month=u'08', count=376)
Row(year=u'1972', month=u'05', count=376)
…
Row(year=u'1993', month=u'11', count=1)
Row(year=u'1952', month=u'11', count=1)
Row(year=u'2005', month=u'12', count=1)
Row(year=u'1999', month=u'12', count=1)
```

The results for assignment 2 are the same when using regular sql queries.

3)

```
Row(year=u'2006', month=u'07', station=u'83270', avg=31.143548387096775)
Row(year=u'1969', month=u'08', station=u'62080', avg=31.137096774193555)
Row(year=u'1996', month=u'08', station=u'78140', avg=31.133870967741938)
Row(year=u'2001', month=u'07', station=u'62400', avg=31.132258064516133)
…
Row(year=u'1990', month=u'03', station=u'63450', avg=7.922580645161292)
Row(year=u'1975', month=u'10', station=u'126300', avg=7.922580645161289)
Row(year=u'1999', month=u'04', station=u'127230', avg=7.921666666666668)
…
```

4)

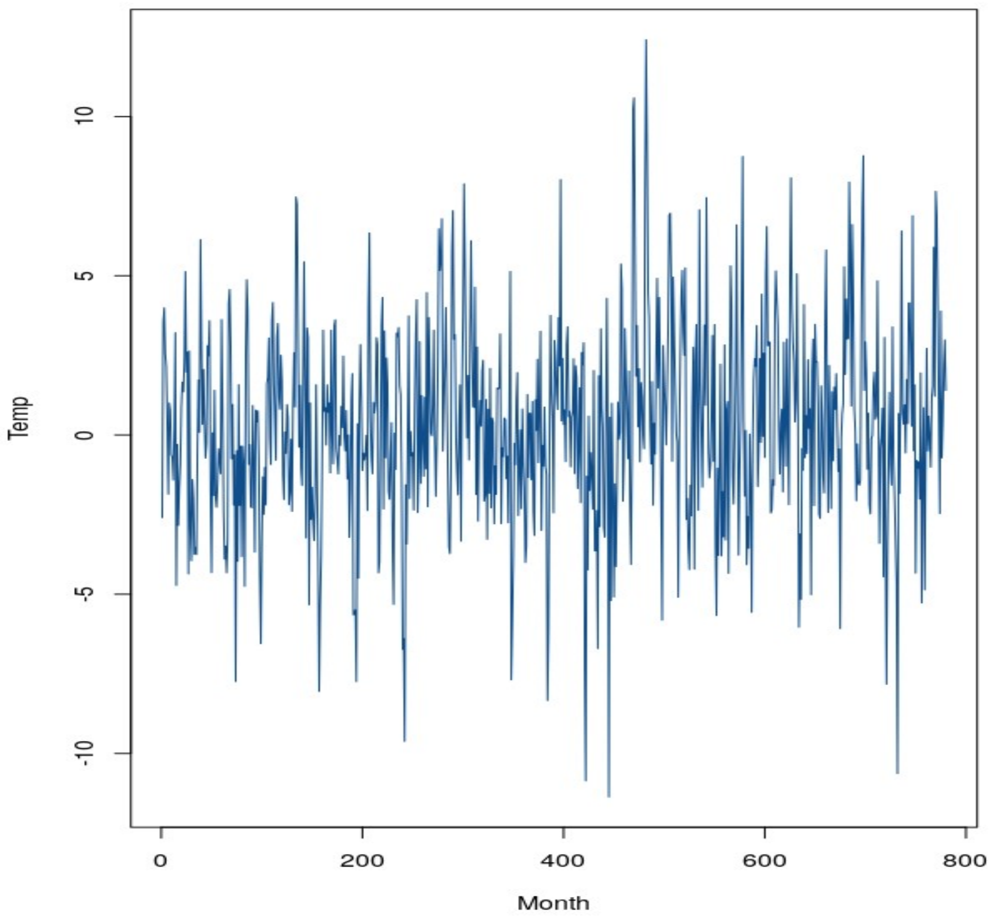
This result is empty.

5)

```
Row(year=u'2015', month=u'07', value=28.0235294117647)
Row(year=u'2006', month=u'08', value=26.132352941176464)
Row(year=u'2008', month=u'08', value=24.444117647058803)
Row(year=u'2000', month=u'07', value=23.976470588235284)
Row(year=u'2015', month=u'09', value=23.835294117647056)
…
```

6)

```
Row(month=u'05', year=u'1953', diff=0.3323871489246102)
Row(month=u'04', year=u'1953', diff=2.8280173813641554)
Row(month=u'03', year=u'1952', diff=-4.362502918517487)
Row(month=u'02', year=u'1952', diff=2.6063966489308124)
Row(month=u'01', year=u'1952', diff=1.9753551989113909)
```



Appendix

1a

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


sc = SparkContext()
sqlContext = SQLContext(sc)

rdd = sc.textFile("/user/x_krisi/data/temperature-readings.csv")
parts = rdd.map(lambda x: x.split(";"))
temp = parts.map(lambda x: Row(station = x[0], date = x[1],
        year = x[1].split("-")[0], time=x[2], value=float(x[3]),
    quality=x[4]))
schemaTemp = sqlContext.createDataFrame(temp)
schemaTemp.registerTempTable("temp")

schemaTemp = schemaTemp.filter(schemaTemp['year'] >= 1950)
schemaTemp = schemaTemp.filter(schemaTemp['year'] <= 2014)


min_temps = schemaTemp.groupBy('year').agg(
    F.min('value').alias(
    'value'))

min_temps = min_temps.join(schemaTemp, ['year', 'value']).select(
    'year', 'station', 'value').orderBy(
    F.desc("value"))

max_temps = schemaTemp.groupBy('year').agg(
    F.max('value').alias(
    'value'))

max_temps = max_temps.join(schemaTemp, ['year', 'value']).select(
        'year', 'station', 'value').orderBy(
        F.desc("value"))


min_temps.rdd.saveAsTextFile("lab2_res/min_temps")
max_temps.rdd.saveAsTextFile("lab2_res/max_temps")
```

2 api

```
schemaTemp.registerTempTable("temp")
schemaTemp = schemaTemp.filter(schemaTemp['year'] >= 1950)
schemaTemp = schemaTemp.filter(schemaTemp['year'] <= 2014)
schemaTemp = schemaTemp.filter(schemaTemp['value'] >= 10)

count_temps = schemaTemp.groupBy(
         'year','month').agg(
         F.count('value').alias(
         'value')).orderBy(
         F.desc('value'))
count_distinct = schemaTemp.select(
    'station','year','month').distinct().groupBy(
    "year","month").count().orderBy(F.desc('count'))

count_temps.rdd.saveAsTextFile("lab2_res/count_temps")
count_distinct.rdd.saveAsTextFile("lab2_res/count_distinct")
```

2 regular sql

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


sc = SparkContext()
sqlContext = SQLContext(sc)

rdd = sc.textFile("/user/x_krisi/data/temperature-readings.csv")
parts = rdd.map(lambda x: x.split(";"))

temp = parts.map(lambda x: Row(station = x[0], date = x[1],
        year = x[1].split("-")[0],month = x[1].split("-")[1],
        time=x[2], value=float(x[3]),
        quality=x[4]))

schemaTemp = sqlContext.createDataFrame(temp)
schemaTemp.registerTempTable("temp")

count_temps = sqlContext.sql("SELECT year, month, count(value) as value FROM temp
    where year>=1950 and year<=2014 and value>=10.0
    group by year, month ORDER BY value DESC")

count_distinct = sqlContext.sql("SELECT year, month, count(DISTINCT station) as count FROM
    temp where year>=1950 and year<=2014 and value>=10.0
    group by year, month ORDER BY count DESC")

count_temps.rdd.saveAsTextFile("lab2_res/count_temps_sql")
count_distinct.rdd.saveAsTextFile("lab2_res/count_distinct_sql")
```

3

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


sc = SparkContext()
sqlContext = SQLContext(sc)

rdd = sc.textFile("/user/x_krisi/data/temperature-readings.csv")
parts = rdd.map(lambda x: x.split(";"))
temp = parts.map(lambda x: Row(station = x[0], date = x[1],
          year = x[1].split("-")[0], month = x[1].split("-")[1],
          day = x[1].split("-")[2],time=x[2], value=float(x[3]),
          quality=x[4]))
schemaTemp = sqlContext.createDataFrame(temp)
schemaTemp.registerTempTable("temp")

schemaTemp = schemaTemp.filter(schemaTemp['year'] >= 1960)
schemaTemp = schemaTemp.filter(schemaTemp['year'] <= 2016)

avg_temp = schemaTemp.groupBy(
          'year','month','day','station').agg(
          (F.max('value')+F.min('value')/ 2).alias('value') )

avg_temp = avg_temp.groupBy(
          'year', 'month','station').agg(
           F.avg('value').alias('avg')).orderBy(F.desc('avg'))

avg_temp.rdd.saveAsTextFile("lab2_res/avg_temp")
```

4

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


sc = SparkContext()
sqlContext = SQLContext(sc)

temp_rdd = sc.textFile("/user/x_krisi/data/temperature-readings.csv")
parts = temp_rdd.map(lambda x: x.split(";"))
temp = parts.map(lambda x: Row(station = x[0], value=float(x[3])))
schemaTemp = sqlContext.createDataFrame(temp)
schemaTemp.registerTempTable("temp")

precipitation_rdd = sc.textFile("/user/x_krisi/data/precipitation-readings.csv")
parts = precipitation_rdd.map(lambda x: x.split(";"))
precipitation = parts.map(lambda x: Row(station = x[0],
                                year = x[1].split("-")[0], month = x[1].split("-")[1],
                                day = x[1].split("-")[2],value=float(x[3])))
schemaRain = sqlContext.createDataFrame(precipitation)
schemaRain.registerTempTable("precipitation")

temp_filter = schemaTemp.groupBy('station').agg(F.max('value').alias('max_temp'))
temp_filter = temp_filter.filter(temp_filter['max_temp'] >= 25)
temp_filter = temp_filter.filter(temp_filter['max_temp'] <= 30)


rain_filter = schemaRain.groupBy('station','year','month','day').agg(
        F.sum('value').alias('value'))
rain_filter = rain_filter.groupBy('station').agg(F.max('value').alias('value'))
rain_filter = rain_filter.filter(rain_filter['value'] >= 100)
rain_filter = rain_filter.filter(rain_filter['value'] <= 200)

joined_filter = temp_filter.join(
        rain_filter,['station']).select(
        'station','max_temp','value').orderBy(F.desc('station')).show()
```

5

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


sc = SparkContext()
sqlContext = SQLContext(sc)

station_rdd = sc.textFile("/user/x_krisi/data/stations-Ostergotland.csv")
parts = station_rdd.map(lambda x: x.split(";"))
stations = parts.map(lambda x: Row(station = x[0]))
schemaStation = sqlContext.createDataFrame(stations)
schemaStation.registerTempTable("stations")
length = schemaStation.count()

precipitation_rdd = sc.textFile("/user/x_krisi/data/precipitation-readings.csv")
parts = precipitation_rdd.map(lambda x: x.split(";"))
precipitation = parts.map(lambda x: Row(station = x[0],
                                    year = x[1].split("-")[0], month = x[1].split("-")[1],
                                    day = x[1].split("-")[2],value=float(x[3])))
schemaRain = sqlContext.createDataFrame(precipitation)
schemaRain.registerTempTable("precipitation")

rain_in_ost = schemaRain.filter(schemaRain['year'] >= 1993)
rain_in_ost = rain_in_ost.filter(rain_in_ost['year'] <= 2016)

rain_in_ost = schemaRain.join(
        schemaStation,['station']).select('year', 'month','value')

rain_in_ost = rain_in_ost.groupBy(
        'year','month').agg((F.sum('value')/length).alias('value')).orderBy(F.desc('value'))

rain_in_ost.rdd.saveAsTextFile('lab2_res/rain_in_ost')
```

6

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

sc = SparkContext()
sqlContext = SQLContext(sc)

station_rdd = sc.textFile("/user/x_krisi/data/stations-Ostergotland.csv")
parts = station_rdd.map(lambda x: x.split(";"))
stations = parts.map(lambda x: Row(station = x[0]))
schemaStation = sqlContext.createDataFrame(stations)
schemaStation.registerTempTable("stations")
length = schemaStation.count()

rdd = sc.textFile("/user/x_krisi/data/temperature-readings.csv")
parts = rdd.map(lambda x: x.split(";"))
temp = parts.map(lambda x: Row(station = x[0], date = x[1],
                                year = x[1].split("-")[0], month = x[1].split("-")[1],
                                day = x[1].split("-")[2], value=float(x[3])))
schemaTemp = sqlContext.createDataFrame(temp)
schemaTemp.registerTempTable("temp")

schemaTemp = schemaTemp.join(schemaStation,['station']).select(
    'year','month','value','day','station')

# Long-term avg
long_temp = schemaTemp.filter(schemaTemp['year'] >= 1950)
long_temp = long_temp.filter(long_temp['year'] <= 1980)


long_temp = long_temp.groupBy(
                            'year','month','day','station').agg(
                            (F.max('value')+F.min('value')/ 2).alias('value'))

long_temp = long_temp.groupBy('year', 'month').agg(
          F.avg('value').alias('value'))

long_temp = long_temp.groupBy('month').agg(F.avg('value').alias('value'))

# Avg temp
avg_temp = schemaTemp.filter(schemaTemp['year'] >= 1950)
avg_temp = avg_temp.filter(avg_temp['year'] <= 2014)

avg_temp = avg_temp.groupBy(
                            'year','month','day','station').agg(
                            (F.max('value')+F.min('value')/2).alias('value'))

avg_temp = avg_temp.groupBy(
         'year', 'month').agg(F.avg('value').alias('avg'))

# Difference
joint = avg_temp.join(long_temp, ['month'])
joint = joint.withColumn('diff',joint['avg']-joint['value'])
joint = joint.select(
         'month','year','diff').orderBy(
         F.desc('year'),F.desc('month'))

joint.rdd.repartition(1).saveAsTextFile('lab2_res/joint')
```