732A54 - Big Data Analytics

BDA2 Spark SQL Exercises

Ashraf Sarhan (ashsa762)

December 23, 2016

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.

- `year, station with the max, maxValue ORDER BY maxValue DESC`
- `year, station with the min, minValue ORDER BY minValue DESC`

spark_sql_mmt_extractor.py:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 12 17:06:24 2016

@author: ashraf
"""
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row
from pyspark.sql import functions as sql

iFile = 'data/temperature-readings.csv'
oFile1 = 'data/sql_max_temperature'
oFile2 = 'data/sql_min_temperature'

sc = SparkContext(appName = "MinMaxTempExtractorSparkSQLJob")

sqlContext = SQLContext(sc)

inFile = sc.textFile(iFile).map(lambda line: line.split(";")). \
map(lambda l: Row(station = l[0], year = l[1].split("-")[0], month = l[1].split("-")[1], \
day = l[1].split("-")[2], time = l[2], temp = float(l[3]), quality = l[4]))

tempSchema = sqlContext.createDataFrame(inFile)

tempSchema.registerTempTable("TempSchema")

minTemp = tempSchema.select('year','station','temp').groupBy('year'). \
agg(sql.min('temp').alias("minTemp")).orderBy('minTemp', ascending = [0])

minTemp.coalesce(1).saveAsTextFile(oFile1)

maxTemp = sqlContext.sql("SELECT FIRST(year), FIRST(station), MAX(temp) AS maxTemp \
                          FROM TempSchema \
                          WHERE year >= 1950 AND year <= 2014 \
                          GROUP BY year \
                          ORDER BY maxTemp DESC")

maxTemp.coalesce(1).saveAsTextFile(oFile2)
```

output:

```
Row(year=u'1990', station=u'106500', minTemp=-35.0),
Row(year=u'1952', station=u'108320', minTemp=-35.5),
Row(year=u'1974', station=u'106270', minTemp=-35.6),
Row(year=u'1954', station=u'103090', minTemp=-36.0),
Row(year=u'1992', station=u'107400', minTemp=-36.1),
Row(year=u'1975', station=u'105370', minTemp=-37.0),
Row(year=u'1972', station=u'105360', minTemp=-37.5),
Row(year=u'1995', station=u'103410', minTemp=-37.6),
Row(year=u'2000', station=u'105220', minTemp=-37.6),
Row(year=u'1957', station=u'108170', minTemp=-37.8)
```

output:

```
Row(year=u'1975', station=u'105370', maxTemp=36.1),
Row(year=u'1992', station=u'107400', maxTemp=35.4),
Row(year=u'1994', station=u'105370', maxTemp=34.7),
Row(year=u'2010', station=u'105260', maxTemp=34.4),
Row(year=u'2014', station=u'108110', maxTemp=34.4),
Row(year=u'1989', station=u'105370', maxTemp=33.9),
Row(year=u'1982', station=u'107530', maxTemp=33.8),
Row(year=u'1968', station=u'106270', maxTemp=33.7),
Row(year=u'1966', station=u'103410', maxTemp=33.5),
Row(year=u'1983', station=u'106580', maxTemp=33.3)
```

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.

- `year, month, value ORDER BY value DESC`
- `year, month, value ORDER BY value DESC`

spark_sql_temp_count_extractor.py:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 12 17:06:24 2016

@author: ashraf
"""
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row

iFile = 'data/temperature-readings.csv'
oFile1 = 'data/sql_over_ten_mth_temp_counts'
oFile2 = 'data/sql_over_ten_temp_distinct_counts'


sc = SparkContext(appName = "TempCounterSparkSQLJob")

sqlContext = SQLContext(sc)

inFile = sc.textFile(iFile) \
            .map(lambda line: line.split(";")) \
            .map(lambda l: \
                Row(station = l[0], date = l[1],  \
                    year = l[1].split("-")[0], \
                    month = l[1].split("-")[1], time = l[2], \
                    temp = float(l[3]), quality = l[4]))

tempSchema = sqlContext.createDataFrame(inFile)

tempSchema.registerTempTable("TempSchema")

overTenTemp = sqlContext.sql(" \
                        SELECT FIRST(year) AS year, FIRST(month) AS month, COUNT(temp) AS counts\
                        FROM TempSchema \
                        WHERE temp >= 10 AND year >= 1950 AND year <= 2014\
                        GROUP BY year, month \
                        ORDER BY counts DESC")

overTenTemp.saveAsTextFile(oFile1)

overTenTempDistinct = sqlContext.sql(" \
                        SELECT FIRST(year) AS year, FIRST(month) AS month, FIRST(station) AS station, \
                                COUNT(DISTINCT temp) AS counts\
                        FROM TempSchema \
                        WHERE temp >= 10 AND year >= 1950 AND year <= 2014\
                        GROUP BY year, month, station \
                        ORDER BY counts DESC")


overTenTempDistinct.saveAsTextFile(oFile2)
```

output:

```
Row(year=u'2014', month=u'07', counts=147910),
Row(year=u'2011', month=u'07', counts=147060),
Row(year=u'2010', month=u'07', counts=143860),
Row(year=u'2012', month=u'07', counts=138166),
Row(year=u'2013', month=u'07', counts=134297),
Row(year=u'2009', month=u'07', counts=133570),
Row(year=u'2011', month=u'08', counts=133483),
Row(year=u'2009', month=u'08', counts=129007),
Row(year=u'2013', month=u'08', counts=128920),
Row(year=u'2003', month=u'07', counts=128360)
```

output:

```
Row(year=u'1994', month=u'07', station=u'95170', counts=208),
Row(year=u'1994', month=u'07', station=u'107400', counts=206),
Row(year=u'1994', month=u'07', station=u'86340', counts=206),
Row(year=u'1994', month=u'07', station=u'78300', counts=205),
Row(year=u'1994', month=u'07', station=u'97510', counts=205),
Row(year=u'1994', month=u'07', station=u'86440', counts=205),
Row(year=u'1994', month=u'07', station=u'64560', counts=204),
Row(year=u'1994', month=u'07', station=u'53220', counts=203),
Row(year=u'2014', month=u'07', station=u'92410', counts=202),
Row(year=u'2010', month=u'07', station=u'75250', counts=201)
```

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.

- `year, month, station, avgMonthlyTemperature ORDER BY avgMonthlyTemperature DESC`

spark_sql_temp_avg_extractor.py:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 12 17:06:24 2016

@author: ashraf
"""
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row

iFile = 'data/temperature-readings.csv'
oFile = 'data/sql_station_avg_mth_temp'

sc = SparkContext(appName="AvgTempSparkSQLJob")

sqlContext = SQLContext(sc)

inFile = sc.textFile(iFile) \
            .map(lambda line: line.split(";")) \
            .map(lambda l: \
                Row(station = l[0], date = l[1],  \
                    year = l[1].split("-")[0], \
                    month = l[1].split("-")[1], time = l[2], \
                    temp = float(l[3]), quality = l[4]))

tempSchema = sqlContext.createDataFrame(inFile)
```

```
tempSchema.registerTempTable("TempSchema")

avgMonthlyTemp = sqlContext.sql(" \
                        SELECT FIRST(year) AS year, FIRST(month) AS month, FIRST(station) AS station, \
                                AVG(temp) AS AvgTemp \
                        FROM TempSchema \
                        WHERE year >= 1960 AND year <= 2014 \
                        GROUP BY year, month, station \
                        ORDER BY AvgTemp DESC")

avgMonthlyTemp.saveAsTextFile(oFile)
```

output:

```
Row(year=u'2014', month=u'07', station=u'96000', AvgTemp=26.3),
Row(year=u'1994', month=u'07', station=u'65450', AvgTemp=23.65483870967742),
Row(year=u'1994', month=u'07', station=u'95160', AvgTemp=23.505376344086027),
Row(year=u'1994', month=u'07', station=u'75120', AvgTemp=23.26881720430107),
Row(year=u'1994', month=u'07', station=u'105260', AvgTemp=23.143820224719107),
Row(year=u'1994', month=u'07', station=u'85280', AvgTemp=23.108602150537635),
Row(year=u'1983', month=u'08', station=u'54550', AvgTemp=23.0),
Row(year=u'1975', month=u'08', station=u'54550', AvgTemp=22.9625),
Row(year=u'1994', month=u'07', station=u'96550', AvgTemp=22.957894736842114),
Row(year=u'1994', month=u'07', station=u'96000', AvgTemp=22.931182795698923)]
```

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.

- `station, maxTemp, maxDailyPrecipitation ORDER BY station DESC`

spark_sql_max_temp_prec_extractor.py:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 12 17:06:24 2016

@author: ashraf
"""
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row

iFile = 'data/temperature-readings.csv'
iFile2 = 'data/precipitation-readings.csv'
oFile = 'data/sql_max_temperature_precipitation'

sc = SparkContext(appName="MaxTempPrecExtractorSparkSQLJob")
```

```
sqlContext = SQLContext(sc)

# Temperatures
inFile = sc.textFile(iFile) \
            .map(lambda line: line.split(";")) \
            .map(lambda l: \
                Row(station = l[0], \
                    date = l[1],  \
                    year = l[1].split("-")[0], \
                    month = l[1].split("-")[1], \
                    day = l[1].split("-")[2], \
                    time = l[2], \
                    temp = float(l[3]), \
                    quality = l[4]))

tempSchema = sqlContext.createDataFrame(inFile)

tempSchema.registerTempTable("TempSchema")

# precipitation
inFile2 = sc.textFile(iFile2) \
            .map(lambda line: line.split(";")) \
            .map(lambda l: \
                Row(station = l[0], \
                    date = l[1],  \
                    year = l[1].split("-")[0], \
                    month = l[1].split("-")[1], \
                    day = l[1].split("-")[2], \
                    time = l[2], \
                    prec = float(l[3]), \
                    quality = l[4]))

precSchema = sqlContext.createDataFrame(inFile2)

precSchema.registerTempTable("PrecSchema")

tempPrec = sqlContext.sql(" \
                SELECT t.station, \
                        MAX(t.temp) AS maxTemp, \
                        MAX(p.dailyPrec) AS maxDailyPrec\
                FROM TempSchema t, \
                    (SELECT station, SUM(prec) as dailyPrec \
                    FROM PrecSchema \
                    GROUP BY station, day) p \
                WHERE t.station = p.station AND \
                        t.temp >= 25 AND t.temp <= 30 AND \
                        p.dailyPrec >= 100 AND p.dailyPrec <= 200 \
                GROUP BY t.station")

tempPrec.saveAsTextFile(oFile)
```

output:

```
Row(station=u'147560', maxTemp=29.9, maxDailyPrec=177.89999999999975),
Row(station=u'68560', maxTemp=28.5, maxDailyPrec=153.19999999999996),
Row(station=u'63160', maxTemp=29.9, maxDailyPrec=186.69999999999996),
Row(station=u'84310', maxTemp=30.0, maxDailyPrec=116.8),
Row(station=u'167710', maxTemp=29.6, maxDailyPrec=193.39999999999944),
Row(station=u'107140', maxTemp=30.0, maxDailyPrec=197.59999999999948),
Row(station=u'183750', maxTemp=29.4, maxDailyPrec=188.9999999999996),
Row(station=u'66110', maxTemp=26.3, maxDailyPrec=153.09999999999985),
Row(station=u'127130', maxTemp=29.9, maxDailyPrec=193.19999999999965),
Row(station=u'96190', maxTemp=30.0, maxDailyPrec=194.9999999999995)
```

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 monthly precipitation for each station before calculating the monthly average (by averaging over stations).

- `station, maxTemp, maxDailyPrecipitation ORDER BY station DESC`

spark_sql_ostergot_avg_prec_extractor.py:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 12 17:06:24 2016

@author: ashraf
"""
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row

iFile = 'data/stations-Ostergotland.csv'
iFile2 = 'data/precipitation-readings.csv'
oFile = 'data/OstergotlandAveMonthlyPrec'

sc = SparkContext(appName="OstergotlandAvgMonthlyPrecSparkSQLJob")

sqlContext = SQLContext(sc)


# Ostergotland Stations
ostergotlandStations = sc.textFile(iFile) \
            .map(lambda line: line.split(";")) \
            .map(lambda l: int(l[0])) \
            .distinct().collect()

isOstergotlandStation = (lambda s: s in ostergotlandStations)

inFile = sc.textFile(iFile2) \
            .map(lambda line: line.split(";")) \
            .filter(lambda l: isOstergotlandStation(int(l[0]))) \
            .map(lambda l: \
                Row(station = l[0], \
                    date = l[1],  \
                    year = l[1].split("-")[0], \
                    month = l[1].split("-")[1], \
                    day = l[1].split("-")[2], \
                    time = l[2], \
                    prec = float(l[3]), \
                    quality = l[4]))

precSchema = sqlContext.createDataFrame(inFile)

precSchema.registerTempTable("PrecSchema")

avgPrec = sqlContext.sql(" \
                    SELECT ps.year AS year, ps.month AS month, AVG(prec.totPrec) AS avgMonthPrec \
                    FROM PrecSchema ps, \
                        (SELECT year, month, station, SUM(prec) AS totPrec \
                        FROM PrecSchema  \
                        GROUP BY year, month, station) AS prec \
                    WHERE ps.station = prec.station AND \
                           ps.year = prec.year AND \
                            ps.month = prec.month AND \
                            ps.year >= 1993 AND ps.year <= 2016 \
                    GROUP BY ps.year, ps.month")

avgPrec = avgPrec.rdd.repartition(1) \
                .sortBy(ascending = False, keyfunc = lambda \
                    (year, month, prec): (year, month))

avgPrec.saveAsTextFile(oFile)
```

output:

```
Row(year=u'2006', month=u'12', avgMonthPrec=29.75475174118164),
Row(year=u'1995', month=u'01', avgMonthPrec=26.000000000000004),
Row(year=u'1995', month=u'02', avgMonthPrec=31.799999999999613),
Row(year=u'1995', month=u'03', avgMonthPrec=34.40000000000035),
Row(year=u'1995', month=u'04', avgMonthPrec=61.69999999999967),
Row(year=u'2013', month=u'01', avgMonthPrec=21.572687760779043),
Row(year=u'1995', month=u'05', avgMonthPrec=26.000000000000004),
Row(year=u'2007', month=u'01', avgMonthPrec=68.67610803324125),
Row(year=u'2013', month=u'02', avgMonthPrec=25.812057286626647),
Row(year=u'1995', month=u'06', avgMonthPrec=97.19999999999871)
```

6. Compare the average monthly temperature (find the difference) in the period 1950-2014 for all stations in Östergotland with long-term monthly averages in the period of 1950-1980.

- `Year, month, difference ORDER BY year DESC, month DESC`

spark_ostergot_avg_temp_diff_extractor.py:

output: