Lab 2: BDA2 - Spark SQL- Exercises

Anubhav Dikshit(anudi287) and Sae Won Jun (saeju204)

2019-05-02

Contents

Question 1:

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

Code:

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

iFile = 'Data/temperature-readings.csv'
oFile1 = 'Data/sql_max_temperature'
oFile2 = 'Data/sql_min_temperature'
fromYear = 1950
toYear = 2014

sc = SparkContext(appName = "MinMaxTempExtractorSparkSQLJob_anudi287")

sqlContext = SQLContext(sc)

inFile = sc.textFile(iFile) \
            .map(lambda line: line.split(";")) \
            .filter(lambda obs:
                               (int(obs[1][:4]) >= fromYear and
                                int(obs[1][:4]) <= toYear)) \
            .map(lambda obs: \
                Row(station = obs[0], date = obs[1],  \
                    year = obs[1].split("-")[0], time = obs[2],
                    temp = float(obs[3]), quality = obs[4]))

tempSchema = sqlContext.createDataFrame(inFile)

tempSchema.registerTempTable("TempSchema")

"""
Q1.1 year, station with the max, maxValue ORDER BY maxValue DESC
"""
maxTemp = sqlContext.sql(
        """
        SELECT year, MAX(temp) AS temp
        FROM TempSchema
        GROUP BY year
        ORDER BY temp DESC
        """
    )

maxTemp.rdd.repartition(1) \
             .sortBy(ascending=False, keyfunc=lambda (year, temp): temp)

maxTemp.take(10)

maxTemp.write.save(oFile1)

"""
Q2.1 year, station with the min, minValue ORDER BY minValue DESC
"""
minTemp = sqlContext.sql(
        """
        SELECT year, MIN(temp) AS temp
        FROM TempSchema
        GROUP BY year
        ORDER BY temp DESC
        """
    )

minTemp.rdd.repartition(1) \
             .sortBy(ascending=False, keyfunc=lambda (year, temp): temp)

minTemp.take(10)

minTemp.write.save(oFile2)
```

Copying code from local

```
scp .\serial_code.py x_anudi@heffa.nsc.liu.se:/nfshome/x_anudi/
```

Running the spark code

```
./runYarn.sh spark_max_temp_prec_extractor.py

# copy files to hdfs from local
hdfs dfs -copyFromLocal temperature-readings.csv Data/
hdfs dfs -copyFromLocal stations-Ostergotland.csv Data/


# copy output from hdfs to local
hdfs dfs -copyToLocal /user/x_anudi/Data/over_ten_mth_temp_counts
hdfs dfs -copyToLocal /user/x_anudi/Data/over_ten_temp_distinct_counts
```

Output Max Temp:

```
maxTemp.take(10)
[Row(year=u'1975', station=u'86200', temp=36.1),
Row(year=u'1992', station=u'63600', temp=35.4),
Row(year=u'1994', station=u'117160', temp=34.7),
Row(year=u'2014', station=u'96560', temp=34.4),
Row(year=u'2010', station=u'75250', temp=34.4),
Row(year=u'1989', station=u'63050', temp=33.9),
Row(year=u'1982', station=u'94050', temp=33.8),
Row(year=u'1968', station=u'137100', temp=33.7),
Row(year=u'1966', station=u'151640', temp=33.5),
Row(year=u'2002', station=u'78290', temp=33.3)]

maxTemp2.take(10)
[Row(year=u'1975', temp=36.1),
Row(year=u'1992', temp=35.4),
Row(year=u'1994', temp=34.7),
Row(year=u'2010', temp=34.4),
Row(year=u'2014', temp=34.4),
Row(year=u'1989', temp=33.9),
Row(year=u'1982', temp=33.8),
Row(year=u'1968', temp=33.7),
Row(year=u'1966', temp=33.5),
Row(year=u'1983', temp=33.3)]
```

Output Min Temp:

```
minTemp.take(10)
[Row(year=u'1990', station=u'147270', temp=-35.0),
Row(year=u'1952', station=u'192830', temp=-35.5),
Row(year=u'1974', station=u'166870', temp=-35.6),
Row(year=u'1954', station=u'113410', temp=-36.0),
Row(year=u'1992', station=u'179960', temp=-36.1),
Row(year=u'1975', station=u'157860', temp=-37.0),
Row(year=u'1972', station=u'167860', temp=-37.5),
Row(year=u'1995', station=u'182910', temp=-37.6),
Row(year=u'2000', station=u'169860', temp=-37.6),
Row(year=u'1957', station=u'159970', temp=-37.8)]

minTemp2.take(10)
[Row(year=u'1990', temp=-35.0),
Row(year=u'1952', temp=-35.5),
Row(year=u'1974', temp=-35.6),
Row(year=u'1954', temp=-36.0),
Row(year=u'1992', temp=-36.1),
Row(year=u'1975', temp=-37.0),
Row(year=u'1972', temp=-37.5),
Row(year=u'1995', temp=-37.6),
Row(year=u'2000', temp=-37.6),
Row(year=u'1957', temp=-37.8)]
```

Question 2:

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

Code:

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

iFile = 'Data/temperature-readings.csv'
oFile = 'Data/sql_over_ten_temp_distinct_counts'


sc = SparkContext(appName = "TempCounterSparkSQLJob_anudi287")

sqlContext = SQLContext(sc)

inFile = sc.textFile(iFile) \
            .map(lambda line: line.split(";")) \
            .map(lambda obs: \
                Row(station = obs[0], date = obs[1],  \
                    year = obs[1].split("-")[0], \
                    month = obs[1].split("-")[1], time = obs[2], \
                    yymm = obs[1][:7], \
                    temp = float(obs[3]), quality = obs[4]))

tempSchema = sqlContext.createDataFrame(inFile)
```

```
tempSchema.registerTempTable("TempSchema")

"""
Q1. Temperatures readings higher than 10 degrees
"""
overTenTemp = sqlContext.sql(" \
                        SELECT FIRST(year), FIRST(month), COUNT(temp) AS counts\
                        FROM TempSchema \
                        WHERE temp >= 10 AND year >= 1950 AND year <= 2014\
                        GROUP BY year, month \
                        ORDER BY counts DESC")

"""
Q2. Distinct Temperatures readings higher than 10 degrees
"""
overTenTempDistinct = tempSchema.filter(tempSchema["temp"] > 10) \
                                .groupBy("yymm") \
                                .agg(F.countDistinct("station").alias("count"))


overTenTempDistinct = overTenTempDistinct.rdd.repartition(1) \
                            .sortBy(ascending = False, keyfunc = lambda \
                                    (yymm, counts): counts)

overTenTempDistinct.write.save(oFile)
```

Output Distinct Temperatures readings counts:

```
print overTenTempDistinct.take(10)
[Row(yymm=u'1972-10', count=378),
 Row(yymm=u'1973-05', count=377),
 Row(yymm=u'1973-06', count=377),
 Row(yymm=u'1973-09', count=376),
 Row(yymm=u'1972-08', count=376),
 Row(yymm=u'1972-05', count=375),
 Row(yymm=u'1972-06', count=375),
 Row(yymm=u'1972-09', count=375),
 Row(yymm=u'1971-08', count=375),
 Row(yymm=u'1972-07', count=374)]
```

Question 3:

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

Code:

```
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_anudi287")

sqlContext = SQLContext(sc)

inFile = sc.textFile(iFile) \
            .map(lambda line: line.split(";")) \
                       .filter(lambda obs:
                               (int(obs[1][:4]) >= 1960 and
                                int(obs[1][:4]) <= 2014)) \
                       .map(lambda obs: Row(station=int(obs[0]),
                                            day=obs[1],
                                            month=obs[1][:7],
                                            temp=float(obs[3])))

tempSchema = sqlContext.createDataFrame(inFile)

tempSchema.registerTempTable("TempSchema")

avgMonthTemp = sqlContext.sql(
        """
        SELECT mytbl.month, mytbl.station, AVG(mytbl.max_temp + mytbl.min_temp) / 2 AS avg_temp
        FROM
        (
        SELECT month, station, MIN(temp) AS min_temp, MAX(temp) AS max_temp
        FROM TempSchema
        GROUP BY day, month, station
        ) AS mytbl
        GROUP BY mytbl.month, mytbl.station
        ORDER BY AVG(mytbl.max_temp + mytbl.min_temp) / 2 DESC
        """
    )

avgMonthTemp.rdd.repartition(1).sortBy(ascending=False,
                                keyfunc=lambda (month, station, temp): temp)

avgMonthTemp.write.save(oFile)
```

Output Average monthly temperatures:

```
print avgMonthTemp.take(10)
[Row(month=u'2014-07', station=96000, avg_temp=26.3),
Row(month=u'1994-07', station=96550, avg_temp=23.07105263157895),
Row(month=u'1983-08', station=54550, avg_temp=23.0),
Row(month=u'1994-07', station=78140, avg_temp=22.970967741935485),
Row(month=u'1994-07', station=85280, avg_temp=22.872580645161293),
Row(month=u'1994-07', station=75120, avg_temp=22.858064516129033),
Row(month=u'1994-07', station=65450, avg_temp=22.856451612903232),
Row(month=u'1994-07', station=96000, avg_temp=22.808064516129033),
Row(month=u'1994-07', station=95160, avg_temp=22.764516129032256),
Row(month=u'1994-07', station=86200, avg_temp=22.711290322580645)]
```

Question 4:

- station, maxTemp, maxDailyPrecipitation ORDER BY station DESC

Code:

```
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_anudi287")

sqlContext = SQLContext(sc)

# Temperatures
temperature_data = sc.textFile(iFile)
temperature_obs = temperature_data.map(lambda line: line.split(";")) \
                                  .map(lambda obs: Row(station=int(obs[0]),
                                                       temp=float(obs[3])))
schema_temp_readings = sqlContext.createDataFrame(temperature_obs)
schema_temp_readings.registerTempTable("temp_readings")

# precipitation
precipitation_data = sc.textFile(iFile2)
precipitation_obs = precipitation_data.map(lambda line: line.split(";")) \
                                      .map(lambda obs: Row(station=int(obs[0]),
                                                           day=obs[1],
                                                           precip=float(obs[3])))
schema_precip_readings = sqlContext.createDataFrame(precipitation_obs)
schema_precip_readings.registerTempTable("precip_readings")

combined = sqlContext.sql(
        """
        SELECT tr.station, MAX(temp) AS max_temp, MAX(precip) AS max_precip
        FROM temp_readings AS tr
        INNER JOIN
        (
        SELECT station, SUM(precip) AS precip
        FROM precip_readings
        GROUP BY day, station
        ) AS pr
        ON tr.station = pr.station
        WHERE (temp >= 25 AND temp <= 30)
        AND (precip >= 100 AND precip <= 200)
        GROUP BY tr.station
        ORDER BY tr.station DESC
        """
        )

tempPrec = combined.rdd.repartition(1) \
        .sortBy(ascending=False, keyfunc=lambda (station, temp, precip): station)

tempPrec.take(10)

tempPrec.saveAsTextFile(oFile)
```

Output Max daily temperatures/precipitation:

```
tempPrec.take(10)
[Row(station=97510, max_temp=30.0, max_precip=103.99999999999999),
Row(station=75250, max_temp=30.0, max_precip=101.8),
Row(station=71420, max_temp=30.0, max_precip=106.3),
Row(station=52350, max_temp=30.0, max_precip=101.6)]
```

Question 5:

- station, maxTemp, maxDailyPrecipitation ORDER BY station DESC

Code:

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

iFile = 'Data/stations-Ostergotland.csv'
iFile2 = 'Data/precipitation-readings.csv'
oFile = 'Data/sql_OstergotlandAveMonthlyPrec'

sc = SparkContext(appName="OstergotlandAvgMonthlyPrecSparkSQLJob_anudi287")

sqlContext = SQLContext(sc)

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

ostergotlandStations = sc.broadcast(ostergotlandStations)

ostergotlandStations = {station: True for station in ostergotlandStations.value}

precipitations = sc.textFile(iFile2).map(lambda line: line.split(";")) \
                                         .filter(lambda obs:
                                                   ostergotlandStations.get(int(obs[0]), False)) \
                                         .map(lambda obs: Row(day=obs[1],
                                                               month=obs[1][:7],
                                                               station=int(obs[0]),
                                                               precip=float(obs[3])))

precSchema = sqlContext.createDataFrame(precipitations)
precSchema.registerTempTable("PrecSchema")

avgMthPrec = sqlContext.sql(
        """
        SELECT mytbl2.month, AVG(mytbl2.precip) AS avg_precip
        FROM
        (
        SELECT mytbl1.month, mytbl1.station, SUM(mytbl1.precip) AS precip
        FROM
        (
        SELECT month, station, SUM(precip) AS precip
        FROM PrecSchema
        GROUP BY day, month, station
        ) AS mytbl1
        GROUP BY mytbl1.month, mytbl1.station
        ) AS mytbl2
        GROUP BY mytbl2.month
        ORDER BY mytbl2.month DESC
        """
    )

avgMthPrec.rdd.repartition(1).sortBy(ascending=False, keyfunc=lambda (month, precip): month)

avgMthPrec.saveAsTextFile(oFile)
```

Output Ostergotland average monthly precipitation:

```
print avgMthPrec.take(10)
[Row(month=u'2016-07', avg_precip=0.0),
 Row(month=u'2016-06', avg_precip=47.6625),
 Row(month=u'2016-05', avg_precip=29.250000000000004),
 Row(month=u'2016-04', avg_precip=26.9),
 Row(month=u'2016-03', avg_precip=19.9625),
 Row(month=u'2016-02', avg_precip=21.5625),
 Row(month=u'2016-01', avg_precip=22.325),
 Row(month=u'2015-12', avg_precip=28.925),
 Row(month=u'2015-11', avg_precip=63.887499999999996),
 Row(month=u'2015-10', avg_precip=2.2625)]
```

Question 6:

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

Code:

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

iFile = 'Data/stations-Ostergotland.csv'
iFile2 = 'Data/temperature-readings.csv'
oFile = 'Data/sql_OstergotlandAveMonthlyDiffTemp'

sc = SparkContext(appName="OstergotlandAvgMonthlyTempDiffSparkSQLJob_anudi287")

sqlContext = SQLContext(sc)

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

ostergotlandStations = sc.broadcast(ostergotlandStations)

ostergotlandStations = {station: True for station in ostergotlandStations.value}

temperatures = sc.textFile(iFile2) \
            .map(lambda line: line.split(";")) \
            .filter(lambda obs: ostergotlandStations.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], \
                    day = obs[1].split("-")[2], \
                    yymm = obs[1][:7], \
                    yymmdd = obs[1], \
                    time = obs[2], \
                    temp = float(obs[3]), \
                    quality = obs[4]))

tempSchema = sqlContext.createDataFrame(temperatures)
tempSchema.registerTempTable("TempSchema")

avgMthTemp = sqlContext.sql("""
        SELECT one.yymm,
            AVG(one.minTemp + one.maxTemp) / 2 AS avgTemp
        FROM
        (
        SELECT yymm,
                year,
                yymmdd,
                MIN(temp) AS minTemp,
                MAX(temp) AS maxTemp
        FROM TempSchema
        GROUP BY yymmdd,
                    yymm,
                    year,
                    station
        ) AS one
        WHERE one.year >= 1950 AND one.year <= 2014
        GROUP BY one.yymm
        """)


#note changes
longTermAvgTemp = avgMthTemp.filter(avgMthTemp.substring(avgMthTemp["yymm"], 1, 4) <= 1980) \
                    .groupBy(avgMthTemp.substring(avgMthTemp["yymm"], 6, 7).alias("month")) \
                    .agg(avgMthTemp.avg(avgMthTemp["avgTemp"]).alias("longTermAvgTemp"))

diffTemp = avgMthTemp.join(longTermAvgTemp,
                                (avgMthTemp.substring(avgMthTemp["yymm"], 6, 7) ==
                                  longTermAvgTemp["month"]), "inner")

diffTemp = diffTemp.select(diffTemp["yymm"],
                          (diffTemp.abs(diffTemp["avgTemp"]) -
                             diffTemp.abs(diffTemp["longTermAvgTemp"])).alias("diffTemp"))

diffTemp = diffTemp.rdd.repartition(1).sortBy(ascending = False,
                             keyfunc = lambda (yymm, diff): yymm)

diffTemp.write.save(oFile)
```

Output Ostergotland average monthly precipitation temperature difference:

```
print diffTemp.take(10)
[Row(yymm=u'2014-12', diffTemp=-0.7938517834097853),
 Row(yymm=u'2014-11', diffTemp=2.0635396726928987),
 Row(yymm=u'2014-10', diffTemp=1.521957490617976),
 Row(yymm=u'2014-09', diffTemp=0.06105818643722749),
 Row(yymm=u'2014-08', diffTemp=-0.6426470719706963),
 Row(yymm=u'2014-07', diffTemp=2.1059218387139893),
 Row(yymm=u'2014-06', diffTemp=-1.8073686197315233),
 Row(yymm=u'2014-05', diffTemp=0.26719065014070154),
 Row(yymm=u'2014-04', diffTemp=2.0661931589915437),
 Row(yymm=u'2014-03', diffTemp=3.176498950234642)]
```