BDA2 - Spark SQL - Exercises

Andreas Christopoulos Charitos (andch552) & Phillip H??lscher (phiho267)

11 5 2019

Contents

BDA2 - Spark - Exercises **2**

Assignments 2

Assignments 1 2

Assignments 1a 2

Assignments 2 4

Assignments 3 6

Assignments 4 7

Assignments 5 9

Assignments 6 11

BDA2 - Spark - Exercises

Every .txt output will be attached as separate file.

In this set of exercises you will work exclusively with Spark. This means that in your programs, you only need to create the **SparkContext** .

In a number of exercises you will be asked to calculated temperature averages (daily and monthly). These are not always computed according to the standard definition of average. In this domain the daily average temperature is calculated by averaging the daily measured maximum and the daily measured minimum temperatures. The monthly average is calculated by averaging the daily maximums and minimums for that month. For example, to get the monthly average for October, take maximums and minimums for each day, sum them up and divide by 62 (which is the same as taking the daily averages, summing them up and divide by the number of days).

There are two ways to write queries in Spark SQL - using built-in API functions or running SQL-like queries. *To pass this lab, you need to*: - (1) use built-in API functions for all the questions - (2) write a regular SQL query for the second question

For each question include the following data in the report and sort it as shown:

- 1. year, station with the max, maxValue ORDER BY maxValue DESC year, station with the min, minValue ORDER BY minValue DESC
- 2. year, month, value ORDER BY value DESC year, month, value ORDER BY value DESC
- 3. year, month, station, avgMonthlyTemperature ORDER BY avgMonthlyTemperature DESC
- 4. station, maxTemp, maxDailyPrecipitation ORDER BY station DESC Note: The correct result for this question should be empty.
- 5. year, month, avgMonthlyPrecipitation ORDER BY year DESC, month DESC
- 6. year, month, difference ORDER BY year DESC, month DESC

Assignments

Assignments 1

- 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.

Assignments 1a

- a) Extend the program to include the station number (*not the station name*) where the maximum/minimum temperature wasmeasured.

Code:

```
# here come the code for assigment 1a - max
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row
from pyspark.sql import functions as F
sc =SparkContext()
sqlContext=SQLContext(sc)

temps= sc.textFile("/user/x_phiho/data/temperature-readings.csv")
```

```
parts = temps.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],  Te
#Inferring the schema and registering the DataFrame as atable
schemaTempReadings =  sqlContext.createDataFrame(tempReadings)
schemaTempReadings.registerTempTable("tempReadings")

max_temps=schemaTempReadings.select(["year","station","Temp"]).filter((schemaTempReadings['year'] <=201
.groupby(['year','station']).agg(F.max('Temp').alias("max_temp")).orderBy('max_temp',ascending=False)

min_temps=schemaTempReadings.select(["year","station","Temp"]).filter((schemaTempReadings['year'] <=201
.groupby(['year','station']).agg(F.min('Temp').alias("min_temp")).orderBy('min_temp',ascending=False)

max_temps_rdd = max_temps.rdd
min_temps_rdd = min_temps.rdd

max_temps_rdd.saveAsTextFile("results/BDA2_1_max")
min_temps_rdd.saveAsTextFile("results/BDA2_1_min")
```

Output - max:

```
# here comes the output - the frist rows
Row(year=u'1975', station=u'86200', max_temp=36.1)
Row(year=u'1975', station=u'95160', max_temp=35.8)
Row(year=u'1975', station=u'96550', max_temp=35.6)
Row(year=u'1975', station=u'106100', max_temp=35.5)
Row(year=u'1992', station=u'63600', max_temp=35.4)
Row(year=u'1975', station=u'75240', max_temp=35.4)
Row(year=u'1992', station=u'63050', max_temp=35.2)
Row(year=u'1975', station=u'96350', max_temp=35.0)
Row(year=u'1975', station=u'96030', max_temp=35.0)
Row(year=u'1992', station=u'85040', max_temp=35.0)
Row(year=u'1992', station=u'76000', max_temp=35.0)
Row(year=u'1975', station=u'97390', max_temp=35.0)
Row(year=u'1992', station=u'75240', max_temp=35.0)
```

Output:

```
# here comes the output - the last rows
Row(year=u'1999', station=u'181900', min_temp=-47.0)
Row(year=u'1966', station=u'192830', min_temp=-47.0)
Row(year=u'1987', station=u'123480', min_temp=-47.3)
Row(year=u'1966', station=u'181900', min_temp=-47.5)
Row(year=u'1978', station=u'155940', min_temp=-47.7)
Row(year=u'1966', station=u'189780', min_temp=-48.0)
Row(year=u'1999', station=u'191910', min_temp=-48.7)
Row(year=u'1999', station=u'191720', min_temp=-48.8)
Row(year=u'1999', station=u'192830', min_temp=-49.0)
Row(year=u'1966', station=u'166870', min_temp=-49.3)
Row(year=u'1966', station=u'179950', min_temp=-49.4)
```

Assignments 2

- 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

Code:

```
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row
from pyspark.sql import functions as F
sc =SparkContext()
sqlContext=SQLContext(sc)


temp=sc.textFile("/user/x_phiho/data/temperature-readings.csv")
parts=temp.map(lambda a: a.split(';'))
tempReadings=parts.map(lambda x: Row(station=x[0], date=x[1], year=int(x[1].split("-")[0]), month=int(x
schemaTempReadings=sqlContext.createDataFrame(tempReadings)

schemaTempReadings.registerTempTable("tempReadingsTable")

df_sql=sqlContext.sql('SELECT year, month, count(station) as value FROM tempReadingsTable WHERE year>=1

df_sql_distinct=sqlContext.sql('SELECT year, month, count(distinct station) as value FROM tempReadingsT

df_sql=df_sql.rdd
df_sql_distinct=df_sql_distinct.rdd

df_sql.saveAsTextFile("results/BDA2_2SQL_results2")
df_sql_distinct.saveAsTextFile("results/BDA2_2SQL_distinct_results2")
```

We had a problem to save the results as rdd and used the function "show()" to print the outout of the script. Below you can see the output of df_sql_distinct show(). Output:

```
  File "BDA2_2SQL_combined.py", line 19, in <module>
    df_sql=df_sql.rdd
AttributeError: 'NoneType' object has no attribute 'rdd'

+----+-----+------+
|year|month| value|
+----+-----+------+
|2014|    7|147681|
|2011|    7|146656|
|2010|    7|143419|
|2012|    7|137477|
|2013|    7|133657|
|2009|    7|133008|
|2011|    8|132734|
|2009|    8|128349|
|2013|    8|128235|
|2003|    7|128133|
|2002|    7|127956|
|2006|    8|127622|
|2008|    7|126973|
|2002|    8|126073|
|2005|    7|125294|
|2011|    6|125193|
|2012|    8|125037|
|2006|    7|124794|
|2010|    8|124417|
|2014|    8|124045|
+----+-----+------+
only showing top 20 rows

+----+-----+-----+
|year|month|value|
+----+-----+-----+
|1972|   10|  378|
|1973|    6|  377|
|1973|    5|  377|
|1973|    9|  376|
|1972|    8|  376|
|1972|    9|  375|
|1972|    6|  375|
|1971|    8|  375|
|1972|    5|  375|
|1972|    7|  374|
|1971|    6|  374|
|1971|    9|  374|
|1971|    5|  373|
|1973|    8|  373|
|1974|    8|  372|
|1974|    6|  372|
|1970|    8|  370|
|1974|    9|  370|
|1973|    7|  370|
|1974|    5|  370|
+----+-----+-----+
only showing top 20 rows
```

Assignments 3

- 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

Code:

```
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row
from pyspark.sql import functions as F
sc =SparkContext()
sqlContext=SQLContext(sc)

temps= sc.textFile("/user/x_phiho/data/temperature-readings.csv")
parts = temps.map(lambda l:l.split(";"))
tempReadings = parts.map(lambda p: Row(station=p[0],  date=p[1], year=p[1].split("-")[0],month=p[1].spl
#Inferring the schema and registering the DataFrame as atable
schemaTempReadings =  sqlContext.createDataFrame(tempReadings)
schemaTempReadings.registerTempTable("tempReadings")


schemaAvg=schemaTempReadings.select('year','month','day','station','Temp')\
.filter((schemaTempReadings['year'] <=2014) & (schemaTempReadings["year"]>=1960 )).groupby( "year","mon
.agg( (F.avg('Temp') ).alias("daily_avg_temp"))\
.select('year','month','station','daily_avg_temp').groupBy('year','month','station')\
.agg( (F.avg('daily_avg_temp') ).alias("avg_temperature"))\
.orderBy('year','month','station',ascending=False)

schemaAvg_rdd = schemaAvg.rdd

schemaAvg_rdd.saveAsTextFile("results/BDA2_3_results")
```

Output:

```
# results of the first rows
Row(year=u'2014', month=u'12', station=u'99450', avg_temperature=1.9897849462365587)
Row(year=u'2014', month=u'12', station=u'99280', avg_temperature=2.3321236559139775)
Row(year=u'2014', month=u'12', station=u'99270', avg_temperature=2.434301075268817)
Row(year=u'2014', month=u'12', station=u'98490', avg_temperature=-1.0755376344086023)
Row(year=u'2014', month=u'12', station=u'98290', avg_temperature=0.5602150537634408)
Row(year=u'2014', month=u'12', station=u'98230', avg_temperature=0.4267473118279569)
Row(year=u'2014', month=u'12', station=u'98210', avg_temperature=0.6279569892473118)
Row(year=u'2014', month=u'12', station=u'98180', avg_temperature=0.13225806451612887)
Row(year=u'2014', month=u'12', station=u'98040', avg_temperature=0.5327956989247311)
Row(year=u'2014', month=u'12', station=u'97530', avg_temperature=-1.6580837173579106)
Row(year=u'2014', month=u'12', station=u'97510', avg_temperature=-1.1068548387096777)
Row(year=u'2014', month=u'12', station=u'97400', avg_temperature=-0.7415322580645163)
Row(year=u'2014', month=u'12', station=u'97370', avg_temperature=-1.4209167410429686)
```

Assignments 4

- 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 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

Code:

```
# create the setup
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row
from pyspark.sql import functions as F
sc =SparkContext()
sqlContext=SQLContext(sc)


# path to temperature-readings-tiny.csv
path_unifolder_temp_tiny = "/user/x_phiho/data/temperature-readings.csv"
path_unifolder_precip_tiny = "/user/x_phiho/data/precipitation-readings.csv"


# read and create rdd, this needs to be tranfsormed to data frame later on
temps = sc.textFile(path_unifolder_temp_tiny)
parts_temps = temps.map(lambda l:l.split(";"))
tempReadings = parts_temps.map(lambda p: Row(station=p[0],  date=p[1], year=p[1].split("-")[0], month=p


# read and create rdd, this needs to be tranfsormed to data frame later on
precip = sc.textFile(path_unifolder_precip_tiny)
parts_precip = precip.map(lambda l:l.split(";"))
precipReadings = parts_precip.map(lambda p: Row(station=p[0],  date=p[1], year=p[1].split("-")[0], mont


# create a data frame from rdd
schemaTempReadings =  sqlContext.createDataFrame(tempReadings)
schemaTempReadings.registerTempTable("tempReadings")


# create a data frame from rdd
schemaPrecipReadings =  sqlContext.createDataFrame(precipReadings)
schemaPrecipReadings.registerTempTable("precipReadings")


# find the max temp for each station
schemaTempReadings_max = schemaTempReadings\
.select(["station", "Temp"])\
.groupBy("station").agg(F.max("Temp").alias("Temp_max"))\
.orderBy("Temp_max", ascending = False)



schemaTempReadings_max_filterd = schemaTempReadings_max\
.select(["station", "Temp_max"])\
.filter((schemaTempReadings_max["Temp_max"] >= 25) & (schemaTempReadings_max["Temp_max"] <=30))


# take just the variables we need
schemaPrecipReadings = schemaPrecipReadings\
.select(["station", "year", "month", "day", "Precip"])



# compute the max measured daily precip
# we need to group by year, month, day and sum up all Precip for this condition
schemaPrecipReadings_sum = schemaPrecipReadings\
.groupBy("station", "year", "month", "day")\
.agg(F.sum("Precip").alias("Precip_daily_sum"))\
.orderBy(["station", "year", "month", "day"], ascending = False)



# find the max for each station
schemaPrecipReadings_station_max = schemaPrecipReadings_sum\
.select(["station", "Precip_daily_sum"])\
.groupBy("station").agg(F.max("Precip_daily_sum").alias("maxDailyPrecipitation"))\
.orderBy(["station"], ascending = False)



# filter for precip restrictions
schemaPrecipReadings_station_max_filter = schemaPrecipReadings_station_max\
.filter((schemaPrecipReadings_station_max["maxDailyPrecipitation"] > 100) &
       (schemaPrecipReadings_station_max["maxDailyPrecipitation"] < 200))



# now we have to join the two data frames together
joined_df = schemaTempReadings_max_filterd\
.join(schemaPrecipReadings_station_max_filter, 'station', 'inner')\
.orderBy('station', ascending=False)


joined_df_rdd = joined_df.rdd
joined_df_rdd.saveAsTextFile("results/BDA2_4_results_updated")
```

Output:

The result (joined_df_rdd) is empty.

Assignments 5

- 5) Calculate the average monthly precipitation for the ??stergotland 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

Code:

```
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row
from pyspark.sql import functions as F
sc =SparkContext()
sqlContext=SQLContext(sc)


precipitation = sc.textFile("/user/x_phiho/data/precipitation-readings.csv")

parts=precipitation.map(lambda a: a.split(';'))
precReadingsRow=parts.map(lambda x: Row(station=x[0], date=x[1], year=int(x[1].split("-")[0]), month=in
schemaPrecReadings=sqlContext.createDataFrame(precReadingsRow)

rdd = sc.textFile("/user/x_phiho/data/stations-Ostergotland.csv")

parts=rdd.map(lambda a: a.split(';'))
statOstRow=parts.map(lambda x: Row(station=x[0], name=x[1]))
schemaStatOst=sqlContext.createDataFrame(statOstRow)

schemaPrecReadings=schemaPrecReadings.filter( (schemaPrecReadings['year'] >= 1993) & (schemaPrecReading

schemaPrecReadings=schemaPrecReadings.join(schemaStatOst, 'station', 'inner')

schemaPrecReadings=schemaPrecReadings.groupBy('station', 'year', 'month').agg(F.sum('precip')).groupBy(

schemaPrecReadings_rdd = schemaPrecReadings.rdd

schemaPrecReadings_rdd.saveAsTextFile("results/BDA2_5_results")
```

Output:

```
  File "BDA2_5.py", line 26, in <module>
    schemaPrecReadings_rdd = schemaPrecReadings.rdd
AttributeError: 'NoneType' object has no attribute 'rdd'

+----+-----+-------------------------+
|year|month|avg_monthly_precipitation|
+----+-----+-------------------------+
|2016|    7|                      0.0|
|2016|    6|                  47.6625|
|2016|    5|       29.250000000000004|
|2016|    4|       26.90000000000001|
|2016|    3|      19.962500000000002|
|2016|    2|                 21.5625|
|2016|    1|                  22.325|
|2015|   12|      28.924999999999997|
|2015|   11|       63.88750000000002|
|2015|   10|                  2.2625|
|2015|    9|      101.29999999999998|
|2015|    8|      26.987499999999997|
|2015|    7|      119.09999999999997|
|2015|    6|       78.66250000000002|
|2015|    5|       93.22499999999998|
|2015|    4|      15.337499999999999|
|2015|    3|       42.61250000000001|
|2015|    2|      24.824999999999996|
|2015|    1|       59.11250000000003|
|2014|   12|       35.46250000000001|
+----+-----+------------------------+
only showing top 20 rows
```

Assignments 6

- 6) Compare the average monthly temperature (find the difference) in the period 1950-2014 for all stations in ??stergotland with long-term monthly averages in the period of 1950-1980. Make a plot of your results. HINT: The first step is to find the monthly averages for each station. Then, you can average over all stations to acquire the average temperature for a specific year and month. This RDD/Data Frame can be used to compute the long-term average by averaging over all the years in theinterval. The output should contain the following information: Year, month, difference Code:

```
# create the setup
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row
from pyspark.sql import functions as F
sc =SparkContext()
sqlContext=SQLContext(sc)

from pyspark.sql.functions import col, round, upper
from  pyspark.sql.functions import abs

path_unifolder_temp_tiny = "/user/x_phiho/data/temperature-readings.csv"
path_unifolder_stations = "/user/x_phiho/data/stations-Ostergotland.csv"

# read and create rdd, this needs to be tranfsormed to data frame later on
temps = sc.textFile(path_unifolder_temp_tiny)
parts_temps = temps.map(lambda l:l.split(";"))
tempReadings = parts_temps.map(lambda p: Row(station=p[0],  date=p[1],
                                             year=p[1].split("-")[0],
                                             month=p[1].split("-")[1],
                                             day=p[1].split("-")[2],
                                             time=p[2],  Temp=float(p[3]),
                                             quality=p[4]))

# create a data frame from rdd
schemaTempReadings =  sqlContext.createDataFrame(tempReadings)
schemaTempReadings.registerTempTable("tempReadings")


# read and create rdd, this needs to be tranfsormed to data frame later on
stations = sc.textFile(path_unifolder_stations)
# create DataFrame from RDD
parts_stations = stations.map(lambda a: a.split(';'))
schemaStationsOestergotland = parts_stations.map(lambda x: Row(station = x[0], name = x[1]))
schemaStationsOestergotland = sqlContext.createDataFrame(schemaStationsOestergotland)

# join to filter for ostg stations
schemaTempReadings = schemaTempReadings.join(schemaStationsOestergotland, 'station', 'inner')

# filter for given time interval
tempReadings_filtered_2014 = schemaTempReadings\
.select(["station", "year", "month", "day", "Temp"])\
.filter((schemaTempReadings["year"] >= 1950) & (schemaTempReadings["year"] <= 2014))

# to calculate the monthly avg for each station do we need the max and min temp for
# each month and station
schemaTempReadings_max_min_2014 = tempReadings_filtered_2014.groupBy("year", "month", "station")\
.agg(F.max("Temp").alias("Temp_max"),F.min("Temp").alias("Temp_min"))

# function to calculate the avg temp as in the exercise asked for
def avg_temp(max_temp, min_temp):
    avg = (max_temp + min_temp)/2.0
    return avg

# compute the avg monthly temp for each station
schemaTempReadings_avg_2014 = schemaTempReadings_max_min_2014\
.withColumn("Temp_avg",
            round(avg_temp(schemaTempReadings_max_min_2014["Temp_max"],
                           schemaTempReadings_max_min_2014["Temp_min"]),1))


# sort the avg temp
schemaTempReadings_avg_ordered_2014 = schemaTempReadings_avg_2014\
.select(["year", "month", "Temp_avg"])\
.orderBy("Temp_avg", ascending = False)


# compute the long term monthly avg

# filter >= 1950 & <= 1980
long_term_month_avg_filterd = schemaTempReadings_avg_ordered_2014\
.filter((schemaTempReadings_avg_ordered_2014["year"] >= 1950) &
          (schemaTempReadings_avg_ordered_2014["year"] <= 1980))


# avg over all month
long_term_month_avg = long_term_month_avg_filterd\
.groupBy("month").agg(round(F.avg("Temp_avg"),1).alias("LT_avg_monthly"))



# function to compute the differences
def differences(Temp_avg, montly_avg):
    return abs(Temp_avg - montly_avg)

# to compute the differences we have to join the data frames first
temp_differ = schemaTempReadings_avg_ordered_2014.join(long_term_month_avg, "month", "inner")


# compute the temp difference
temp_differ = temp_differ.withColumn("differences",
                                     round(differences(temp_differ["Temp_avg"],
                                                 temp_differ["LT_avg_monthly"]),1))\
.orderBy(["year", "month"], ascending = False)


# show the required result
temp_differ_result = temp_differ.select("year", "month", "differences")
```

```
#temp_differ_result_rdd = temp_differ_result.rdd
#temp_differ_result_rdd.saveAsTextFile("results/BDA2_6_results_updated2")

# Printing extract of results.
print(temp_differ_result.show(10))
```

Output:

```
+----+-----+-----------+
|year|month|differences|
+----+-----+-----------+
|2014|   12|        5.1|
|2014|   12|        0.8|
|2014|   12|        3.1|
|2014|   12|        0.1|
|2014|   12|        2.3|
|2014|   12|        3.3|
|2014|   12|        0.4|
|2014|   12|        0.1|
|2014|   12|        0.1|
|2014|   12|        3.4|
+----+-----+-----------+
only showing top 10 rows
```