Lab 2 - Big Data

Spark SQL

johed883 and mikmo937

Notice that the "printed" outputs are fragments of the whole output.

Redo all exercises from BDA1 using Spark SQL whenever possible.

The initial processing of csv files can be done using Spark's map.

There are two ways to write queries in SparkSQL - using built-in API functions or running SQL-like queries. Topass this lab, youneedto use built-in API functions for all the 5 exercises. For each exercise, include the following data in the report and sort it as shown.

Assignment 1

What are the lowest and highest temperatures measured each year for the period 1950-2014.

Sort as

year, station with the max, maxValue ORDER BY maxValue DESC

year, station with the min, minValue ORDER BY minValue DESC

In [ ]:
```
#!/usr/bin/env python3

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

sc = SparkContext()
sqlCon = SQLContext(sc) # obtaining sql content from sc object

# This path is to the file on hdfs
temperature_file = sc.textFile("BDA/input/temperature-readings.csv")

lines = temperature_file.map(lambda line: line.split(";"))

# mapping the data we want
tempReads = lines.map(lambda p : Row(year= int(p[1][0:4]),station=p[0],temp=float(p[3])))

# creating the data frame
schematemp = sqlCon.createDataFrame(tempReads)

schematemp.registerTempTable("year_temps")

# filtering the years

schematemp = schematemp.filter((schematemp.year<=2014) & (schematemp.year >=1950))

# picking out max temp for each year
schematemp_max = schematemp.groupBy('year').agg(F.max('temp').alias('temp'))
# min temp
schematemp_min = schematemp.groupBy('year').agg(F.min('temp').alias('temp'))


# Joining them to get the station in the dataframe
schematemp_max = schematemp.join(schematemp_max,['year','temp'])
schematemp_min = schematemp.join(schematemp_min,['year','temp'])

# selecting to get correct order of the columns then sort
schematemp_max = schematemp_max.select('year','station','temp').orderBy(F.desc('temp'))
schematemp_min = schematemp_min.select('year','station','temp').orderBy(F.desc('temp'))

schematemp_max.rdd.saveAsTextFile("BDA/output/max")
schematemp_min.rdd.saveAsTextFile("BDA/output/min")
```

Output:

min

Row(year=1990, station='147270', temp=-35.0)

Row(year=1990, station='166870', temp=-35.0)

Row(year=1952, station='192830', temp=-35.5)

Row(year=1974, station='179950', temp=-35.6)

Row(year=1974, station='166870', temp=-35.6)

Row(year=1954, station='113410', temp=-36.0)

Row(year=1992, station='179960', temp=-36.1)

Row(year=1975, station='157860', temp=-37.0)

Row(year=1972, station='167860', temp=-37.5)

Row(year=1995, station='182910', temp=-37.6)

Row(year=2000, station='169860', temp=-37.6)

Row(year=1957, station='159970', temp=-37.8)

max

Row(year=1975, station='86200', temp=36.1)

Row(year=1992, station='63600', temp=35.4)

Row(year=1994, station='117160', temp=34.7)

Row(year=2014, station='96560', temp=34.4)

Row(year=2010, station='75250', temp=34.4)

Row(year=1989, station='63050', temp=33.9)

Row(year=1982, station='94050', temp=33.8)

Row(year=1968, station='137100', temp=33.7)

Row(year=1966, station='151640', temp=33.5)

Row(year=2002, station='78290', temp=33.3)

Row(year=2002, station='78290', temp=33.3)

Row(year=1983, station='98210', temp=33.3)

Assignment 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 itappears only once in the count for that month.

Sort as

year, month, value ORDER BY value DESC

year, month, value ORDER BY value DESC

In [ ]:
```
#!/usr/bin/env python3

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

sc = SparkContext()
sqlCon = SQLContext(sc) # obtaining sql content from sc object

# This path is to the file on hdfs
temperature_file = sc.textFile("BDA/input/temperature-readings.csv")

lines = temperature_file.map(lambda line: line.split(";"))

# mapping the data we want
tempReads = lines.map(lambda p : Row(year= int(p[1][0:4]),month=int(p[1][5:7]),temp=float(p[3])))

# creating the data frame
schematemp = sqlCon.createDataFrame(tempReads)

schematemp.registerTempTable("yearmon_temps")

# filtering the years

schematemp = schematemp.filter((schematemp.year<=2014) & (schematemp.year >=1950))

# filter temp

schematemp = schematemp.filter((schematemp.temp>10))

# count year-months
count_first = schematemp.select('year','month').groupBy(['year','month']).count()

# sort by count and change name to value

count_first = count_first.orderBy(F.desc('count'))

count_first.rdd.saveAsTextFile("BDA/output/first")


# mapping the data we want
tempReads = lines.map(lambda p : Row(year= int(p[1][0:4]),month=int(p[1][5:7]),station=p[0],temp=float(p[3])))

# creating the data frame
schematemp = sqlCon.createDataFrame(tempReads)

schematemp.registerTempTable("yearmon_temps")

# filtering the years

schematemp = schematemp.filter((schematemp.year<=2014) & (schematemp.year >=1950))

# filter temp

schematemp = schematemp.filter((schematemp.temp>10))

# pickoing ot unique values for year month and station
count_sec = schematemp.select('year','month','station').distinct()

# count year-months
count_sec = count_sec.groupBy(['year','month']).count()

# sort by count and change name to value, also remove station from output

count_sec = count_sec.select('year','month','count').orderBy(F.desc('count'))

count_sec.rdd.saveAsTextFile("BDA/output/sec")
```

Output:

first part

Row(year=2014, month=7, count=147681)

Row(year=2011, month=7, count=146656)

Row(year=2010, month=7, count=143419)

Row(year=2012, month=7, count=137477)

Row(year=2013, month=7, count=133657)

Row(year=2009, month=7, count=133008)

Row(year=2011, month=8, count=132734)

Row(year=2009, month=8, count=128349)

Row(year=2013, month=8, count=128235)

Row(year=2003, month=7, count=128133)

Row(year=2002, month=7, count=127956)

Row(year=2006, month=8, count=127622)

Row(year=2008, month=7, count=126973)

Row(year=2002, month=8, count=126073)

Row(year=2005, month=7, count=125294)

second part

Row(year=1972, month=10, count=378)

Row(year=1973, month=6, count=377)

Row(year=1973, month=5, count=377)

Row(year=1972, month=8, count=376)

Row(year=1973, month=9, count=376)

Row(year=1972, month=5, count=375)

Row(year=1972, month=9, count=375)

Row(year=1972, month=6, count=375)

Row(year=1971, month=8, count=375)

Row(year=1971, month=6, count=374)

Row(year=1972, month=7, count=374)

Row(year=1971, month=9, count=374)

Row(year=1973, month=8, count=373)

Row(year=1971, month=5, count=373)

Row(year=1974, month=6, count=372)

Assignment 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.

Sort as

year, month, station, avgMonthlyTemperature ORDER BY avgMonthlyTemperature DESC

In [ ]:
```
#!/usr/bin/env python3

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

sc = SparkContext(appName = "exercise 1")
sqlContext = SQLContext(sc)

# This path is to the file on hdfs
temperature_file = sc.textFile("BDA/input/temperature-readings.csv")

# split data
lines = temperature_file.map(lambda line: line.split(";"))

# create table with all necessary variables, with additionally date split up into parts
table = lines.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], temp = float(p[3])))


tempDf = sqlContext.createDataFrame(table)
tempDf.registerTempTable("table")

# filter years to period 1960-2014
temps = tempDf.where((tempDf['year'] >= 1960) & (tempDf['year'] <= 2014))

# aggregate separate min respectively max daily temps
tempsMin = temps.groupBy('station', 'year', 'month', 'day').agg(F.min('temp')).alias('dailymin')
tempsMax = temps.groupBy('station', 'year', 'month', 'day').agg(F.max('temp')).alias('dailymax')

# join on relevant keys
join_temp = tempsMin.join(tempsMax, ['station', 'year', 'month', 'day'], 'inner')

# sum daily temps and group by relevant columns
max_days = join_temp.withColumn('dailysum', join_temp['min(temp)'] + join_temp['max(temp)']).groupBy('station', 'year', 'month').agg(F.sum('dailysum').alias('dailysum'))

# aggregate max day for each month and multiply by two
max_date = join_temp.groupBy('station', 'year', 'month').agg((F.max('day') * 2).alias('maxdays'))

# join summed daily temp table with maximum days per month table
avg_temp = max_date.join(max_days, ['station', 'year', 'month'], 'inner').groupBy('station', 'year', 'month')

# sum temps and days, divide to get average
avg_temp_sum = avg_temp.agg(F.sum('dailysum').alias('total_dailysum'), F.sum('maxdays').alias('total_maxdays'))
avg_temp = avg_temp_sum.withColumn("AvgTemp", avg_temp_sum['total_dailysum'] / avg_temp_sum['total_maxdays'])

# select relevant columns and sort according to the exercise
avg_temp = avg_temp.select('station', 'year', 'month', 'AvgTemp').orderBy('AvgTemp', ascending = 0)

# Save the result
avg_temp.rdd.saveAsTextFile("BDA/output")

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder
avg_temp.rdd.saveAsTextFile("BDA/output")
```

Output:

Row(station='78140', year='1994', month='07', AvgTemp=22.970967741935485)

Row(station='85280', year='1994', month='07', AvgTemp=22.872580645161293)

Row(station='75120', year='1994', month='07', AvgTemp=22.85806451612903)

Row(station='65450', year='1994', month='07', AvgTemp=22.856451612903225)

Row(station='96000', year='1994', month='07', AvgTemp=22.80806451612904)

Row(station='95160', year='1994', month='07', AvgTemp=22.76451612903226)

Row(station='86200', year='1994', month='07', AvgTemp=22.711290322580645)

Row(station='78140', year='2002', month='08', AvgTemp=22.700000000000003)

Row(station='76000', year='1994', month='07', AvgTemp=22.698387096774198)

Row(station='78140', year='1997', month='08', AvgTemp=22.666129032258063)

Row(station='105260', year='1994', month='07', AvgTemp=22.65967741935485)

Row(station='76530', year='2006', month='07', AvgTemp=22.598387096774204)

Row(station='86330', year='1994', month='07', AvgTemp=22.548387096774196)

Row(station='75120', year='2006', month='07', AvgTemp=22.527419354838703)

Row(station='54300', year='1994', month='07', AvgTemp=22.469354838709677)

Assignment 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.

Sort as

station, maxTemp, maxDailyPrecipitation ORDER BY station DESC

In [ ]:
```
#!/usr/bin/env python3

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

sc = SparkContext()
sqlCon = SQLContext(sc) # obtaining sql content from sc object

# This path is to the file on hdfs
temperature_file = sc.textFile("BDA/input/temperature-readings.csv")

precipitation_file = sc.textFile("BDA/input/precipitation-readings.csv")

lines = temperature_file.map(lambda line: line.split(";"))

lines_pre = precipitation_file.map(lambda line: line.split(";"))

# mapping the data we want, station as int so we can order by it
tempReads = lines.map(lambda p : Row(station =int(p[0]),temp=float(p[3])))

precipitation = lines_pre.map(lambda x:Row(station=int(x[0]),rain=float(x[3])))

# creating the data frame
schematemp = sqlCon.createDataFrame(tempReads)

schematemp.registerTempTable("yearmon_temps")

schemapre = sqlCon.createDataFrame(precipitation)

schemapre.registerTempTable('pre_station')

# filtering the temps

schematemp = schematemp.filter((schematemp.temp<=30) & (schematemp.temp >=25))

# filter the rain

schemapre = schemapre.filter((schemapre.rain>=100)&(schemapre.rain <=200))

# picking out max for each station

schematemp_max = schematemp.groupBy('station').agg(F.max('temp').alias('maxTemp'))

schemapre_max = schemapre.groupBy('station').agg(F.max('rain').alias('maxDailyPrecipitation'))

# joining the dataframes on station

merge = schematemp_max.join(schemapre_max,schematemp_max['station'] == schemapre_max['station'],'inner')

# sorting
#merge = merge.orderBy(F.desc('station'))

merge.rdd.saveAsTextFile("BDA/output/first")
```

The output from the previous code is empty as there are no matching stations in the two dataset after filtering on temperature between 25-30 and precipitation between 100-200mm.

Assignment 5

Calculate the average monthly precipitation for the Östergotland region (list of stations is provided in the separate file) for the period 1993-2016. In orderto dothis, you willfirstneed 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.

Sort as

year, month, avgMonthlyPrecipitation ORDER BY year DESC, month DESC

In [ ]:
```
#!/usr/bin/env python3

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

sc = SparkContext(appName = "exercise 1")
sqlContext = SQLContext(sc)

# This path is to the file on hdfs
# read station data and split
station_file = sc.textFile("BDA/input/stations-Ostergotland.csv")
s_lines = station_file.map(lambda line: line.split(";"))
s_table = s_lines.map(lambda p: Row(stationID = p[0]))  # select id

stationsDf = sqlContext.createDataFrame(s_table)
stationsDf.registerTempTable("s_table")


# reading precipitation data and split on relevant columns
precipitation_file = sc.textFile("BDA/input/precipitation-readings.csv")
p_lines = precipitation_file.map(lambda line: line.split(";"))
p_table = p_lines.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], precip = float(p[3])))


precipDf = sqlContext.createDataFrame(p_table)
precipDf.registerTempTable("p_table")

# join the dataframes on station ID to sort out only those in Östergötland
precip = stationsDf.join(precipDf, stationsDf["stationID"] == precipDf["station"], 'inner')

# filter to years 1993-2016
precip = precip.where((precip['year'] >= 1993) & (precip["year"] <= 2016))

# calculate average precipitation by year and month
avg_prec = precip.groupBy('year', 'month').agg(F.avg('precip').alias('avgMonthlyPrecipitation'))

# sort accordingly
avg_prec = avg_prec.orderBy(["year", "month"], ascending=[0, 0])

# Save the result
avg_prec.rdd.saveAsTextFile("BDA/output")
```

Output:

Row(year='2016', month='07', avgMonthlyPrecipitation=0.0)

Row(year='2016', month='06', avgMonthlyPrecipitation=0.07006615214994487)

Row(year='2016', month='05', avgMonthlyPrecipitation=0.04157043879907623)

Row(year='2016', month='04', avgMonthlyPrecipitation=0.03951524054351819)

Row(year='2016', month='03', avgMonthlyPrecipitation=0.028538241601143686)

Row(year='2016', month='02', avgMonthlyPrecipitation=0.03391663389697209)

Row(year='2016', month='01', avgMonthlyPrecipitation=0.032012905538627005)

Row(year='2015', month='12', avgMonthlyPrecipitation=0.04152907394113425)

Row(year='2015', month='11', avgMonthlyPrecipitation=0.09301182893539581)

Row(year='2015', month='10', avgMonthlyPrecipitation=0.0032018397311162215)

Row(year='2015', month='09', avgMonthlyPrecipitation=0.14667873303167414)

Row(year='2015', month='08', avgMonthlyPrecipitation=0.038253012048192765)

Row(year='2015', month='07', avgMonthlyPrecipitation=0.16872675757039127)

Row(year='2015', month='06', avgMonthlyPrecipitation=0.11523530488921442)

Row(year='2015', month='05', avgMonthlyPrecipitation=0.13079621185548926)