732A54: Big Data Analytics - Lab BDA2 – Spark SQL

Umamaheswarababu Maddela (umama339)
Dinesh Sundaramoorthy (dinsu875)

Assignment-1:

Code:

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

sc = SparkContext(appName = "Assignment1")
sqlContext = SQLContext(sc)

# Load a text file and convert each line to a Row.
temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
parts = temperature_file.map(lambda line: line.split(";"))
tempReadings = parts.map(lambda p: Row(station=p[0], date=p[1], year=p[1].\
                split("-")[0], time=p[2], temp=float(p[3]), quality=p[4]))
schemaTempReadings = sqlContext.createDataFrame(tempReadings)
schemaTempReadings.registerTempTable("tempReadings")

schemaTemp = schemaTempReadings.filter((schemaTempReadings.year >= 1950) & (schemaTempReadings.year <= 2014))

# To get maximum temperatures
schemaTemp_Max = schemaTemp.groupBy('year').agg(F.max('temp').alias('temp'))
schemaTemp_Max = schemaTemp_Max.join(schemaTemp, ['year','temp'], 'inner').select('year', 'station', 'temp').\
                orderBy('temp', ascending=False)
schemaTemp_Max.show()
# To get minimum temperatures
schemaTemp_Min = schemaTemp.groupBy('year').agg(F.min('temp').alias('temp'))
schemaTemp_Min = schemaTemp_Min.join(schemaTemp, ['year','temp'], 'inner').\
                select('year', 'station', 'temp').orderBy('temp', ascending=False)
schemaTemp_Min.show()
```

Output:

Maximum Temperatures

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

```
+----+-------+----+
|year|station|temp|
+----+-------+----+
|1975|  86200|36.1|
|1992|  63600|35.4|
|1994| 117160|34.7|
|2014|  96560|34.4|
|2010|  75250|34.4|
|1989|  63050|33.9|
|1982|  94050|33.8|
|1968| 137100|33.7|
|1966| 151640|33.5|
|2002|  78290|33.3|
|2002|  78290|33.3|
|1983|  98210|33.3|
|1970| 103080|33.2|
|1986|  76470|33.2|
|2000|  62400|33.0|
|1956| 145340|33.0|
|1959|  65160|32.8|
|2006|  75240|32.7|
|1991| 137040|32.7|
|1988| 102540|32.6|
+----+-------+----+
only showing top 20 rows
```

Minimum Temperatures

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

```
+----+-------+-----+
|year|station| temp|
+----+-------+-----+
|1990| 147270|-35.0|
|1990| 166870|-35.0|
|1952| 192830|-35.5|
|1974| 166870|-35.6|
|1974| 179950|-35.6|
|1954| 113410|-36.0|
|1992| 179960|-36.1|
|1975| 157860|-37.0|
|1972| 167860|-37.5|
|1995| 182910|-37.6|
|2000| 169860|-37.6|
|1957| 159970|-37.8|
|1983| 191900|-38.2|
|1989| 166870|-38.2|
|1953| 183760|-38.4|
|2009| 179960|-38.5|
|1993| 191900|-39.0|
|1984| 191900|-39.2|
|1984| 123480|-39.2|
|2008| 179960|-39.3|
+----+-------+-----+
only showing top 20 rows
```

Assignment-2:

Count the number of readings for each month in the period of 1950-2014 which are higher than10degrees

Code:

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

sc = SparkContext(appName = "spark")
sqlContext = SQLContext(sc)

#Loading text file and convert each line to a Row
temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
parts = temperature_file.map(lambda line: line.split(";"))

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

schemaTempReadings = sqlContext.createDataFrame(tempReadings)
schemaTempReadings.registerTempTable("tempReadings")

schemaTemp_filter  = schemaTempReadings.filter(schemaTempReadings['temp'] > 10).\
                        filter(schemaTempReadings['year'] >= 1950).filter(schemaTempReadings.year <= 2014)
schemaTemp_count = schemaTemp_filter .groupBy('year','month').agg(F.count('temp').alias('count'))\
                        .orderBy(['count'],ascending=[0])
schemaTemp_count.show()
```

Output:

year, month, value ORDER BY value DESC

```
+----+-----+------+
|year|month| count|
+----+-----+------+
|2014|   07|147681|
|2011|   07|146656|
|2010|   07|143419|
|2012|   07|137477|
|2013|   07|133657|
|2009|   07|133008|
|2011|   08|132734|
|2009|   08|128349|
|2013|   08|128235|
|2003|   07|128133|
|2002|   07|127956|
|2006|   08|127622|
|2008|   07|126973|
|2002|   08|126073|
|2005|   07|125294|
|2011|   06|125193|
|2012|   08|125037|
|2006|   07|124794|
|2010|   08|124417|
|2014|   08|124045|
+----+-----+------+
only showing top 20 rows
```

Taking only distinct readings from each station

Code:

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

sc = SparkContext(appName = "spark")
sqlContext = SQLContext(sc)

#Loading text file and convert each line to a Row
temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
parts = temperature_file.map(lambda line: line.split(";"))

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

schemaTempReadings = sqlContext.createDataFrame(tempReadings)
schemaTempReadings.registerTempTable("tempReadings")

schemaTemp_filter = schemaTempReadings.filter(schemaTempReadings['temp'] > 10)\
                        .filter(schemaTempReadings['year'] >= 1950)\
                        .filter(schemaTempReadings.year <= 2014)
schemaTemp_count = schemaTemp_filter.groupBy('year','month')\
                        .agg(F.countDistinct('station')\
                        .alias('count')).orderBy(['count'],ascending=[0])
schemaTemp_count.show()
```

Output:

year, month, value ORDER BY value DESC

```
+----+-----+-----+
|year|month|count|
+----+-----+-----+
|1972|   10|  378|
|1973|   06|  377|
|1973|   05|  377|
|1973|   09|  376|
|1972|   08|  376|
|1972|   05|  375|
|1971|   08|  375|
|1972|   09|  375|
|1972|   06|  375|
|1972|   07|  374|
|1971|   09|  374|
|1971|   06|  374|
|1971|   05|  373|
|1973|   08|  373|
|1974|   06|  372|
|1974|   08|  372|
|1974|   09|  370|
|1970|   08|  370|
|1971|   07|  370|
|1973|   07|  370|
+----+-----+-----+
only showing top 20 rows
```

Assignment-3:

year, month, station, avgMonthlyTemperature ORDER BY avgMonthlyTemperature DESC

Code:

```
sc = SparkContext(appName = "spark")
sqlContext = SQLContext(sc)

#Loading text file and convert each line to a Row
temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
parts = temperature_file.map(lambda line: line.split(";"))
tempReadings = parts.map(lambda p: Row(station=p[0], date=p[1], year=p[1].split("-")[0], \
               month=p[1].split("-")[1], time=p[2], temp=float(p[3]), quality=p[4]))
schemaTempReadings = sqlContext.createDataFrame(tempReadings)
schemaTempReadings.registerTempTable("tempReadings")

#Filtering
schemaTemp_filter = schemaTempReadings.filter((schemaTempReadings.year >= 1950) & (schemaTempReadings.year <= 2014))

# Selecting daily maximum and minimum temperatures
max_min_temp = schemaTemp_filter.groupBy(['station','date', 'year', 'month']).agg(F.max('temp').alias('max_temp'),\
               F.min('temp').alias('min_temp'))
max_min_temp = max_min_temp.select(['station', 'year', 'month', 'min_temp', 'max_temp'])

# Computing average temperatures
avg_temp = max_min_temp.withColumn('sum', max_min_temp['min_temp'] + max_min_temp['max_temp']).\
           groupBy(['station', 'year','month']).agg(F.avg('sum').alias('temp'))
avg_temp = avg_temp.withColumn('avg_temp', avg_temp['temp']/2).select(['year', 'month', 'station', 'avg_temp']).\
           orderBy('avg_temp', ascending = [0])
avg_temp.show()
```

Output:

```
+----+-----+-------+------------------+
|year|month|station|          avg_temp|
+----+-----+-------+------------------+
|2014|   07|  96000|              26.3|
|1994|   07|  96550|23.071052631578947|
|1983|   08|  54550|              23.0|
|1994|   07|  78140| 22.97096774193548|
|1994|   07|  85280|22.872580645161293|
|1994|   07|  75120|22.858064516129033|
|1994|   07|  65450|22.856451612903225|
|1994|   07|  96000|22.808064516129033|
|1994|   07|  95160| 22.76451612903226|
|1994|   07|  86200| 22.71129032258065|
|2002|   08|  78140|              22.7|
|1994|   07|  76000|22.698387096774194|
|1997|   08|  78140|22.666129032258066|
|1994|   07| 105260|22.659677419354843|
|1975|   08|  54550|22.642857142857142|
|2006|   07|  76530|22.598387096774196|
|1994|   07|  86330|22.548387096774192|
|2006|   07|  75120| 22.52741935483871|
|1994|   07|  54300| 22.46935483870968|
|2006|   07|  78140| 22.45806451612903|
+----+-----+-------+------------------+
only showing top 20 rows
```

Assignment-4:

station, maxTemp, maxDailyPrecipitation ORDER BY station DESC

Code:

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

sc = SparkContext(appName = "Assignment4")
sqlContext = SQLContext(sc)

# Load a temperature-readings file and convert each line to a Row.
temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
parts = temperature_file.map(lambda line: line.split(";"))
tempReadings = parts.map(lambda p: Row(station=p[0], year=p[1].split("-")[0],\
               month=p[1].split("-")[1], time=p[2], temp=float(p[3]), quality=p[4]))
schemaTempReadings = sqlContext.createDataFrame(tempReadings)
schemaTempReadings.registerTempTable("tempReadings")

# Load a precipitation-readings file and convert each line to a Row.
precipitation_file  = sc.textFile("BDA/input/precipitation-readings.csv")
parts = precipitation_file.map(lambda line: line.split(";"))
precipitationReadings = parts.map(lambda p: Row(station=p[0], year=p[1].split("-")[0],\
                        month=p[1].split("-")[1], time=p[2], date=p[1], precipitation=float(p[3]), quality=p[4]))
schemaPrecipitationReadings = sqlContext.createDataFrame(precipitationReadings)
schemaPrecipitationReadings.registerTempTable("precipitationReadings")

#For each station filter the temperatures
schemaTempMax = schemaTempReadings.groupBy('station').agg(F.max('temp').alias('temp'))
schemaTemp = schemaTempMax.filter((schemaTempMax.temp >= 25) & (schemaTempMax.temp <= 30))
schemaTemp = schemaTemp.join(schemaTempReadings, ['station','temp'], 'inner').select('year', 'station', 'temp')

# Get daily precipitation and filtering
schemaPrecDaily = schemaPrecipitationReadings.groupBy('date').sum('precipitation')
schemaPrecDaily = schemaPrecDaily.join(schemaPrecipitationReadings, ['date'], 'inner')
schemaPrecMax = schemaPrecDaily.groupBy('station').agg(F.max('sum(precipitation)').alias('maxDailyPrecipitation'))
schemaPrec = schemaPrecMax.filter ( (schemaPrecMax.maxDailyPrecipitation >= 100) & (schemaPrecMax.maxDailyPrecipitation <= 200) )

schemaTempPrec = schemaPrec.join(schemaTemp, ['station'], 'inner').select('station', 'temp', 'maxDailyPrecipitation').orderBy('station',
ascending=False).show()
```

Output:

```
+-------+----+----------------------+
|station|temp|maxDailyPrecipitation|
+-------+----+----------------------+
+-------+----+----------------------+
```

Assignment-5:

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

Code:

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

#Reading temperatures
precipitation_file  = sc.textFile("BDA/input/precipitation-readings.csv")
parts = precipitation_file.map(lambda l: l.split(";"))
precipitationReadings = parts.map(lambda p: Row(station=p[0], year=p[1].split("-")[0], month=p[1].split("-")[1], time=p[2], date=p[1],
precipitation=float(p[3]), quality=p[4]))
schemaPrecipitationReadings = sqlContext.createDataFrame(precipitationReadings)
schemaPrecipitationReadings.registerTempTable("precipitationReadings")

#Reading Stations
Ostergotland_file = sc.textFile("BDA/input/stations-Ostergotland.csv")
parts = Ostergotland_file.map(lambda line: line.split(";"))
OstergotlandReadings = parts.map(lambda p: Row(station=p[0], station_name =p[1]))
schemaOstergotlandReadings = sqlContext.createDataFrame(OstergotlandReadings)
schemaOstergotlandReadings.registerTempTable("ostergotlandReadings")


## To get only Ostergotland stations
precip_ost = schemaOstergotlandReadings.join(schemaPrecipitationReadings, ['station'])
precip_ost = precip_ost.select('station', 'year', 'month', 'precipitation')\
        .filter(precip_ost['year'] >= 1993).filter(precip_ost.year <= 2016)
precip_ost = precip_ost.groupBy('station', 'year', 'month').agg(F.sum('precipitation').alias('precip'))
precip_ost = precip_ost.select('year', 'month', 'precip')
precip_ost = precip_ost.groupBy('year', 'month').agg(F.avg('precip').alias('average_precip'))\
        .orderBy(['year', 'month'], ascending = False)
precip_ost.show()
```

Output:

```
+----+-----+------------------+
|year|month|    average_precip|
+----+-----+------------------+
|2016|   07|               0.0|
|2016|   06|           47.6625|
|2016|   05|29.250000000000007|
|2016|   04|26.900000000000002|
|2016|   03|19.962500000000002|
|2016|   02|21.562500000000004|
|2016|   01|22.325000000000003|
|2015|   12|28.925000000000004|
|2015|   11| 63.88750000000002|
|2015|   10|            2.2625|
|2015|   09|             101.3|
|2015|   08|26.987500000000004|
|2015|   07|119.09999999999997|
|2015|   06| 78.66250000000001|
|2015|   05|            93.225|
|2015|   04|15.337499999999999|
|2015|   03| 42.61250000000001|
|2015|   02|24.824999999999996|
|2015|   01| 59.11250000000004|
|2014|   12| 35.46250000000001|
+----+-----+------------------+
only showing top 20 rows
```