Baysian Learning 732A54

Lab 2 BDA

Guilherme Barros
guiba484

Eric Herwin
erihe068

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.

Code:

```
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row
from pyspark.sql import functions as F
MinTempYear = (schemaTempReadings.select('year','value').groupBy('year')
               .agg(F.min('value').alias("value")))
joinedMin = MinTempYear.join(schemaTempReadings,
                    ['year', 'value'], 'inner').select('year',
                    'value', 'station').orderBy('value',ascending=False)

print(joinedMin.take(5))
```

a)

Extend the program to include the station number (not the station name) where the maximum/minimum temperature was measured.

Code:

```
MaxTempYear = (schemaTempReadings.select('year','value').groupBy('year')
               .agg(F.max('value').alias("value")))
joinedMin = MaxTempYear.join(schemaTempReadings,
                    ['year', 'value'], 'inner').select('year',
                    'value', 'station').orderBy('value',ascending=False)
print(joinedMax.take(5))
```

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.

Code API:

```
rdd = sc.textFile("path")
parts = rdd.map(lambda l: l.split(";"))
tempReadingsRow = parts.map(lambda p: (p[0], p[1], int(p[1].split("-")[0]),
                                       int(p[1].split("-")[1]),
                                       int(p[1].split("-")[2]), p[2], float(p[3]), p[4] ))
tempReadingsString = ["station", "date", "year", "month", "day","time", "value","quality"]
# Apply the schema to the RDD.
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)
# Register the DataFrame as a table.
schemaTempReadings.registerTempTable("tempReadingsTable")
schemaTempReadings.take(5)
Above10 = (schemaTempReadings.select('year','month','value')
      .filter(schemaTempReadings.value>10).groupBy(['year', 'month'])
      .count().orderBy('count',ascending=False))
Above10distinct = (schemaTempReadings.select('year','month','value')
      .filter(schemaTempReadings.value>10).distinct().groupBy(['year', 'month']).count()
      .orderBy('count',ascending=False))
print(Above10.take(5))
print(Above10distinct.take(5))
```

Code SQL:

```
schemaTempReadings.registerTempTable("tempReadingsTable")
Above10sql = sqlContext.sql("SELECT year, month, count(value) as value FROM tempReadingsTable WHERE valu
Above10distinctsql = sqlContext.sql("SELECT DISTINCT year, month, value FROM tempReadingsTable WHERE val
Above10distinctsql.registerTempTable("Above10distinctsqlTable")
Above10distinctsql2 = sqlContext.sql("SELECT year, month, count(value) as value FROM Above10distinctsqlT
print(Above10sql.take(5))
print(Above10distinctsql2.take(5))
```

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.

Code:

```
MinMonthlyTemperature = (schemaTempReadings.select('year','month','day','station','value')
                         .groupBy('year','month','day','station').agg(F.min('value')
                                                          .alias('dailymin')))
MinMonthlyTemperature = (MinMonthlyTemperature.select('year','month','day','station','dailymin')
                         .groupBy('year','month','station')
                         .agg(F.sum('dailymin').alias('sumdailymin')))
MaxMonthlyTemperature = (schemaTempReadings.select('year','month','day','station','value')
                         .groupBy('year','month','day','station').agg(F.max('value')
                                                          .alias('dailymax')))
MaxMonthlyTemperature = (MaxMonthlyTemperature.select('year','month','day','station','dailymax')
                         .groupBy('year','month','station')
                         .agg(F.sum('dailymax').alias('sumdailymax')))
AvgMonthlyTemperature = (MinMonthlyTemperature.join(MaxMonthlyTemperature,
                                                              ['year', 'month','station'])
                         .select('year','month','station','sumdailymin','sumdailymax') )
finMonthlyTemperature = (AvgMonthlyTemperature
                            .withColumn('dailyavg',
                            (AvgMonthlyTemperature.sumdailymin + AvgMonthlyTemperature.sumdailymax)/62)
                            .orderBy('dailyavg', ascending=False))
print(finMonthlyTemperature.take(5))
```

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.

Code:

```
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row
from pyspark.sql import functions as F
sc = SparkContext()
sqlContext = SQLContext(sc)
rdd = sc.textFile("path")
parts = rdd.map(lambda l: l.split(";"))
tempReadingsRow = parts.map(lambda p: (p[0], p[1], int(p[1].split("-")[0]),
                                       int(p[1].split("-")[1]),
                                       int(p[1].split("-")[2]), p[2], float(p[3]), p[4] ))
tempReadingsString = ["station", "date", "year", "month", "day","time", "value","quality"]
# Apply the schema to the RDD.
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)
rdd_prec = sc.textFile("path")
row_perc = rdd_prec.map(lambda line: line.split(";"))
percReadingsRow = row_perc.map(lambda p: (p[0], p[1], float(p[3])))
percReadingsString = ["station", "date", "value"]
schemaPercReadings = sqlContext.createDataFrame(percReadingsRow, percReadingsString)
#temperature
max_for_stat = (schemaTempReadings.select("station", "date", "value")
                                  .groupBy(["station"]).agg(F.max("value").alias("value")))
filt = max_for_stat.filter(max_for_stat.value > 20).filter(max_for_stat.value < 30)
#percepitation
percep = schemaPercReadings.groupBy(["station", "date"]).agg(F.sum("value").alias("value"))
percep_filt = percep.filter(percep.value > 100).filter(percep.value < 200)
joined = percep_filt.join(filt, "station", "inner").orderBy('station',ascending=False)
print(joined.take(5))
```

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 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).

Code:

```
rdd = sc.textFile(path)
parts = rdd.map(lambda l: l.split(";"))
precReadings = parts.map(lambda p: Row(station=p[0], date=p[1], year=int(p[1].split("-")[0]),
                                       month =  int(p[1].split("-")[1]),
                                       time=p[2],value=float(p[3]), quality=p[4] ))
rdd2 = sc.textFile(path)
parts = rdd2.map(lambda l: l.split(";"))
ostergot_stations = parts.map(lambda p: Row(station=p[0], name=p[1]))
schemaPrecReadings = sqlContext.createDataFrame(precReadings)
ostergot_stations = sqlContext.createDataFrame(ostergot_stations)
PrecInOster = (schemaPrecReadings.join(ostergot_stations, ["station"], 'inner'))
PrecInOster = PrecInOster.select('year','month','station','value').filter(PrecInOster.year>=1993)
TotalPrecInOster = PrecInOster.groupBy('year','month','station').agg(F.sum('value').alias('sumPrec'))
avgPrecInOster = (TotalPrecInOster.select('year','month','sumPrec')
                         .groupBy('year','month').agg(F.avg('sumPrec').alias('avgPrec'))
                  .orderBy(['year', 'month'], ascending=[0,0]) )
print(avgPrecInOster.take(5))
```

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.

Code:

```
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row
from pyspark.sql import functions as F
sc = SparkContext()
sqlContext = SQLContext(sc)
#temperature
rdd = sc.textFile("path")
row = rdd.map(lambda line: line.split(";"))
data_extract = row.map(lambda p: Row(station = p[0],
                                     year = int(p[1].split("-")[0]),
                                     month = int(p[1].split("-")[1]),
                                     value = float(p[3])))
schemaTempReadings = sqlContext.createDataFrame(data_extract)
#stations
rdd_stations = sc.textFile("path")
row_stations = rdd_stations.map(lambda line: line.split(";"))
stationReadingsRow = row_stations.map(lambda p: Row(station = p[0]))
schemastationReadings = sqlContext.createDataFrame(stationReadingsRow)
#filter for year
data_extract = schemaTempReadings.filter(schemaTempReadings.year.between(1950, 1980))
#join to only get ostergotland stations
joined = data_extract.join(schemastationReadings, "station", "left")
#Avg monthly
avgtemp = joined.groupBy("year", "month", "station").agg(F.avg("value").alias("avgtemp")) \
                .groupBy("year", "month").agg(F.avg("avgtemp").alias("avgtemp"))
#Long monthly
longtemp = avgtemp.groupBy("month").agg(F.avg("avgtemp").alias("monthavg"))
#Join
joined2 = avgtemp.join(longtemp, "month")
Final = (joined2.withColumn("difference", (joined2.avgtemp - joined2.monthavg))
         .select("year", "month", "difference").orderBy(['year', 'month'], ascending=[0,0]) )
print(Final.take(5))
```