Mathias Fredholm, matfr953, 961121-8158
David Gumpert Harryson, davha130, 960302-4937

Lab 2 - SparkSQL

Redo the exercises BDA1 using Spark SQL whenever possible. The initial processing of csv files (such as splitting on ;) can be done using Spark's map.

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

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

sc = SparkContext(appName = "Lab2ex1")
sqlContext = SQLContext(sc)

#Importing temperature file and splitting it on ";"
temp_file = sc.textFile("BDA/input/temperature-readings.csv")
temp_lines = temp_file.map(lambda line: line.split(";"))

#Mapping data in rows with appropriate columns
tempReadingsRow = temp_lines.map(lambda p: Row(station=p[0], date=p[1], year=p[1].split("-")[0], time=p[2], value=float(p[3]), quality=p[4]))

#Creating sql dataframe
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow)
schemaTempReadings.registerTempTable("tempReadingsTable")

#Grouping by value
max_temps = schemaTempReadings.select("year", "value").filter((schemaTempReadings["year"]>=1950) & (schemaTempReadings["year"]<=2014)).groupBy("year").
agg(F.max("value").alias("value"))

#Creating stations to join with max_temps since we lose station when selecting on  year and value above
stations = schemaTempReadings.select("year", "station", "value")

#Joining temps with station on year and temperature  and ordering by value (temperature) in descending order
max_temps_station = max_temps.join(stations, ["year", "value"]).orderBy("value", ascending = False)

#Save to output in one file
max_temps_station.rdd.coalesce(1, shuffle=False).saveAsTextFile("BDA/output")
```

Row(year=u'1975', value=36.1, station=u'86200')
Row(year=u'1992', value=35.4, station=u'63600')
Row(year=u'1994', value=34.7, station=u'117160')
Row(year=u'2014', value=34.4, station=u'96560')
Row(year=u'2010', value=34.4, station=u'75250')
Row(year=u'1989', value=33.9, station=u'63050')
Row(year=u'1982', value=33.8, station=u'94050')
Row(year=u'1968', value=33.7, station=u'137100')
Row(year=u'1966', value=33.5, station=u'151640')
Row(year=u'1983', value=33.3, station=u'98210')
Row(year=u'2002', value=33.3, station=u'78290')
Row(year=u'2002', value=33.3, station=u'78290')
Row(year=u'1970', value=33.2, station=u'103080')
Row(year=u'1986', value=33.2, station=u'76470')
Row(year=u'1956', value=33.0, station=u'145340')
Row(year=u'2000', value=33.0, station=u'62400')
Row(year=u'1959', value=32.8, station=u'65160')
Row(year=u'2006', value=32.7, station=u'75240')
Row(year=u'1991', value=32.7, station=u'137040')
Row(year=u'1988', value=32.6, station=u'102540')
Row(year=u'2011', value=32.5, station=u'172770')
Row(year=u'1999', value=32.4, station=u'98210')
Row(year=u'2007', value=32.2, station=u'86420')
Row(year=u'1973', value=32.2, station=u'71470')
Row(year=u'2003', value=32.2, station=u'136420')
Row(year=u'2008', value=32.2, station=u'95130')
Row(year=u'2008', value=32.2, station=u'102390')

Mathias Fredholm, matfr953, 961121-8158
David Gumpert Harryson, davha130, 960302-4937

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

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

sc = SparkContext(appName = "Lab2ex1")
sqlContext = SQLContext(sc)

#Importing temperature file and splitting it on ";"
temp_file = sc.textFile("BDA/input/temperature-readings.csv")
temp_lines = temp_file.map(lambda line: line.split(";"))

#Mapping data in rows with appropriate columns
tempReadingsRow = temp_lines.map(lambda p: Row(station=p[0], date=p[1], year=p[1].split("-")[0], time=p[2], value=float(p[3]), quality=p[4]))

#Creating sql dataframe
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow)
schemaTempReadings.registerTempTable("tempReadingsTable")

#Grouping by value
min_temps = schemaTempReadings.select("year", "value").filter((schemaTempReadings["year"]>=1950) & (schemaTempReadings["year"]<=2014)).groupBy("year").
agg(F.min("value").alias("value"))

#Creating stations to join with max_temps since we lose station when selecting on  year and value above
stations = schemaTempReadings.select("year", "station", "value")

#Joining temps with station on year and temperature  and ordering by value (temperature) in descending order
min_temps_station = min_temps.join(stations, ["year", "value"]).orderBy("value", ascending = False)

#Save to output in one file
min_temps_station.rdd.coalesce(1, shuffle=False).saveAsTextFile("BDA/output")
```

Row(year=u'2008', value=-39.3, station=u'179960')
Row(year=u'2008', value=-39.3, station=u'179960')
Row(year=u'1991', value=-39.3, station=u'179960')
Row(year=u'1973', value=-39.3, station=u'166870')
Row(year=u'2005', value=-39.4, station=u'155790')
Row(year=u'1961', value=-39.5, station=u'181900')
Row(year=u'1964', value=-39.5, station=u'166810')
Row(year=u'1970', value=-39.6, station=u'179950')
Row(year=u'2004', value=-39.7, station=u'166940')
Row(year=u'1988', value=-39.9, station=u'170790')
Row(year=u'1960', value=-40.0, station=u'155910')
Row(year=u'1960', value=-40.0, station=u'160790')
Row(year=u'1960', value=-40.0, station=u'167710')
Row(year=u'1997', value=-40.2, station=u'179960')
Row(year=u'1994', value=-40.5, station=u'179960')
Row(year=u'2006', value=-40.6, station=u'169860')
Row(year=u'2007', value=-40.7, station=u'169860')
Row(year=u'2007', value=-40.7, station=u'169860')
Row(year=u'2013', value=-40.7, station=u'179960')
Row(year=u'1963', value=-41.0, station=u'181900')
Row(year=u'1955', value=-41.2, station=u'160790')
Row(year=u'2003', value=-41.5, station=u'179960')
Row(year=u'1969', value=-41.5, station=u'181900')
Row(year=u'2010', value=-41.7, station=u'191910')
Row(year=u'1996', value=-41.7, station=u'155790')
Row(year=u'1950', value=-42.0, station=u'155910')
Row(year=u'1968', value=-42.0, station=u'179950')
Row(year=u'1951', value=-42.0, station=u'155910')
Row(year=u'1962', value=-42.0, station=u'181900')

Mathias Fredholm, matfr953, 961121-8158
David Gumpert Harryson, davha130, 960302-4937

2. year, month, value ORDER BY value DESC year, month, value ORDER BY value DESC

```
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row
from pyspark.sql import functions as F
sc = SparkContext(appName = "Lab2ex2")
sqlContext = SQLContext(sc)

# Importing neededdnecessary files
temp_file = sc.textFile("BDA/input/temperature-readings.csv")
temp_lines = temp_file.map(lambda line: line.split(";"))

# Mapping data in rows with appropriate columns
tempReadingsRow = temp_lines.map(lambda p: Row(station=p[0], date=p[1], year=p[1].split("-")[0], month=p[1].split("-")[1], time=p[2], value=float(p[3]),
 quality=p[4]))

# Creating a sql data frame
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow)
schemaTempReadings.registerTempTable("tempReadingsTable")

# Getting all temperatures above 10 deg for each month and order by descending temp
high_temps = schemaTempReadings.select("year", "month", "value").filter((schemaTempReadings["year"]>=1950) & (schemaTempReadings["year"]<=2014)).groupBy
("year", "month").agg(F.count(schemaTempReadings["value"]>10).alias("value")).orderBy("value", ascending=False)

# Getting all disctinct station temperatures above 10 deg for each month and order by descending temp
high_dist_temps = schemaTempReadings.select("year", "month", "value", "station").filter((schemaTempReadings["year"]>=1950) & (schemaTempReadings["year"]
<=2014) & (schemaTempReadings["value"] > 10)).select("year", "month", "station").distinct().groupBy("year", "month").agg(F.count("station").alias("value
")).orderBy("value", ascending=False)

# Saving results. Either all or just the distinct.
high_temps.rdd.coalesce(1,shuffle=True).saveAsTextFile("BDA/output")
#high_dist_temps.rdd.coalesce(1,shuffle=True).saveAsTextFile("BDA/output")
```

(a)

Row(year=u'2011', month=u'05', value=157526)
Row(year=u'2013', month=u'01', value=156947)
Row(year=u'2011', month=u'03', value=156928)
Row(year=u'2010', month=u'12', value=156628)
Row(year=u'2012', month=u'10', value=155568)
Row(year=u'2010', month=u'10', value=155547)
Row(year=u'2013', month=u'10', value=155375)
Row(year=u'2010', month=u'03', value=155242)
Row(year=u'2013', month=u'12', value=152620)

(b)

Row(year=u'1972', month=u'10', value=378)
Row(year=u'1973', month=u'06', value=377)
Row(year=u'1973', month=u'05', value=377)
Row(year=u'1972', month=u'08', value=376)
Row(year=u'1973', month=u'09', value=376)
Row(year=u'1976', month=u'05', value=369)
Row(year=u'1970', month=u'06', value=369)
Row(year=u'1970', month=u'09', value=369)
Row(year=u'1975', month=u'09', value=369)
Row(year=u'1970', month=u'07', value=362)
Row(year=u'1974', month=u'07', value=362)
Row(year=u'1967', month=u'09', value=361)
Row(year=u'1972', month=u'07', value=374)
Row(year=u'1971', month=u'09', value=374)
Row(year=u'1971', month=u'06', value=374)
Row(year=u'1968', month=u'05', value=355)

Mathias Fredholm, matfr953, 961121-8158
David Gumpert Harryson, davha130, 960302-4937

Row(year=u'1965', month=u'06', value=355)
Row(year=u'1979', month=u'05', value=354)
Row(year=u'1965', month=u'08', value=354)
Row(year=u'1977', month=u'08', value=354)
Row(year=u'1966', month=u'05', value=354)
Row(year=u'1978', month=u'06', value=354)
Row(year=u'1975', month=u'05', value=367)

3. year, month, station, avgMonthlyTemperature ORDER BY avgMonthlyTemperature DESC

```
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row
from pyspark.sql import functions as F
sc = SparkContext(appName = "Lab2ex3")
sqlContext = SQLContext(sc)

# Importing necessary data
temp_file = sc.textFile("BDA/input/temperature-readings.csv")
temp_lines = temp_file.map(lambda line: line.split(";"))

# Mapping data in rows with appropriate columns
tempReadingsRow = temp_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], time=p[2], value=float(p[3]), quality=p[4]))

# Creating the sql data frame
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow)
schemaTempReadings.registerTempTable("tempReadingsTable")

# Getting the daily min and max temperature
min_max_val_day = schemaTempReadings.select("year", "month", "day", "station", "value").filter((
schemaTempReadings["year"]>=1960) & (schemaTempReadings["year"]<=2014)).groupBy("year", "month",
 "day", "station").agg(F.max(schemaTempReadings["value"]).alias("maxVal"), F.min(schemaTempReadi
ngs["value"]).alias("minVal"))

# for each day getting the avarage temperature (dailyAvg) then grouping over stayear, month and
station to use the avg function to take the monthly avarage
monthly_avg = min_max_val_day.select("year", "month", "station", ((min_max_val_day["maxVal"] + m
in_max_val_day["minVal"])/2).alias("dailyAvg")).groupBy("year", "month", "station").agg(F.avg("d
ailyAvg").alias("monthlyAvg")).orderBy("monthlyAvg", ascending=False)

# Saving the results
monthly_avg.rdd.coalesce(1,shuffle=True).saveAsTextFile("BDA/output")
```

Row(year=u'1999', month=u'08', station=u'78140', monthlyAvg=18.79516129032258)
Row(year=u'1976', month=u'08', station=u'53560', monthlyAvg=18.79516129032258)
Row(year=u'2003', month=u'07', station=u'87450', monthlyAvg=18.793548387096777)
Row(year=u'1966', month=u'07', station=u'53630', monthlyAvg=18.793548387096774)
Row(year=u'2002', month=u'07', station=u'87400', monthlyAvg=18.791935483870965)
Row(year=u'1967', month=u'07', station=u'108110', monthlyAvg=18.79193548387096)
Row(year=u'1980', month=u'07', station=u'161790', monthlyAvg=18.79193548387096)
Row(year=u'1968', month=u'06', station=u'86470', monthlyAvg=18.791666666666668)
Row(year=u'1983', month=u'08', station=u'66500', monthlyAvg=18.790322580645153)
Row(year=u'1973', month=u'07', station=u'140360', monthlyAvg=18.788709677419355)

Mathias Fredholm, matfr953, 961121-8158
David Gumpert Harryson, davha130, 960302-4937

Row(year=u'1972', month=u'07', station=u'64330', monthlyAvg=18.788709677419355)
Row(year=u'1996', month=u'08', station=u'97510', monthlyAvg=18.78870967741935)
Row(year=u'2003', month=u'07', station=u'78290', monthlyAvg=18.78870967741935)
Row(year=u'1966', month=u'07', station=u'108110', monthlyAvg=18.788709677419348)
Row(year=u'1983', month=u'08', station=u'85280', monthlyAvg=18.788709677419348)
Row(year=u'1984', month=u'08', station=u'54390', monthlyAvg=18.787096774193554)
Row(year=u'1995', month=u'07', station=u'54330', monthlyAvg=18.787096774193547)
Row(year=u'1973', month=u'07', station=u'74420', monthlyAvg=18.787096774193547)
Row(year=u'1972', month=u'07', station=u'158750', monthlyAvg=18.787096774193547)
Row(year=u'2010', month=u'07', station=u'82230', monthlyAvg=18.787096774193543)
Row(year=u'1997', month=u'08', station=u'105260', monthlyAvg=18.785999999999998)
Row(year=u'1969', month=u'08', station=u'98400', monthlyAvg=18.783870967741933)
Row(year=u'1973', month=u'07', station=u'72400', monthlyAvg=18.783870967741933)

4. station, maxTemp, maxDailyPrecipitation ORDER BY station DESC

Mathias Fredholm, matfr953, 961121-8158
David Gumpert Harryson, davha130, 960302-4937

```
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 = "Lab2ex3")
sqlContext = SQLContext(sc)

# Importing necessary data and splitting on ";"
temp_file = sc.textFile("BDA/input/temperature-readings.csv")
temp_lines = temp_file.map(lambda line: line.split(";"))
prec_file = sc.textFile("BDA/input/precipitation-readings.csv")
prec_lines = prec_file.map(lambda line: line.split(";"))

# Mapping the data to row format with appropriate columns
tempReadingsRow = temp_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], time=p[2], value=float(p[3])))
precReadingsRow = prec_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], time=p[2], value=float(p[3])))

# Creating the sql data frames
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow)
schemaTempReadings.registerTempTable("tempReadingsTable")
schemaPrecReadings = sqlContext.createDataFrame(precReadingsRow)
schemaPrecReadings.registerTempTable("precReadingsTable")

# Getting the max daily precipitation
daily_prec = schemaPrecReadings.select("year", "month", "day", "station", "value").groupBy
("year", "month", "day", "station").agg(F.sum("value").alias("sumPrec"))

# Getting the max daily temperature
daily_temp = schemaTempReadings.select("year", "month", "day", "station", "value").groupBy
("year", "month", "day", "station").agg(F.max("value").alias("maxTemp"))


# joining the daily max temperatures and max precipitations, then grouping by station to g
et the max for each station over all time
prec_temp = daily_prec.join(daily_temp, ["year", "month", "day", "station"]).select("stati
on", "sumPrec", "maxTemp").groupBy("station").agg(F.max("sumPrec").alias("stationMaxPrec")
, F.max("maxTemp").alias("stationMaxTemp")).orderBy("station", ascending=False)

# Filtering to see what statiots logged a temperature between 25 and 30 and rain between 1
00 and 200.
filtered_max = prec_temp.filter((prec_temp["stationMaxPrec"] >= 100) & (prec_temp["station
MaxPrec"] <= 200) & (prec_temp["stationMaxTemp"] >= 25) & (prec_temp["stationMaxTemp"] <=
30))

# Saving the results
filtered_max.rdd.coalesce(1,shuffle=True).saveAsTextFile("BDA/output")
```

Output: EMPTY

5. year, month, avgMonthlyPrecipitation ORDER BY year DESC, month DESC

Mathias Fredholm, matfr953, 961121-8158
David Gumpert Harryson, davha130, 960302-4937

```
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 = "Lab2ex5")
sqlContext = SQLContext(sc)

# Importing data
station_file = sc.textFile("BDA/input/stations-Ostergotland.csv")
prec_file = sc.textFile("BDA/input/precipitation-readings.csv")

# Splitting data
station_lines = station_file.map(lambda line: line.split(";"))
prec_lines = prec_file.map(lambda line: line.split(";"))

# Getting all relevant stations in row format
stationReadingsRow = station_lines.map(lambda p: Row(station=p[0]))

# Mapping precipitation readings as rows with appropriate columns
precReadingsRow = prec_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], time=p[2], value=float(p[3])))

# Creating sql data frames
schemaStationReadings = sqlContext.createDataFrame(stationReadingsRow)
schemaStationReadings.registerTempTable("stationReadingsTable")
schemaPrecReadings = sqlContext.createDataFrame(precReadingsRow)
schemaPrecReadings.registerTempTable("precReadingsTable")

# Joining precipitation readings with respective station
focal_readings = schemaStationReadings.join(schemaPrecReadings, ["station"])

# Summing up the total precipitation for each station each month then taking the avarage
over all stations for every month
focal_readings = focal_readings.select("year", "month", "station", "value").filter((focal
_readings["year"]>=1993) & (focal_readings["year"]<=2016)).groupBy("year", "month", "stat
ion").agg(F.sum("value").alias("sumMonth")).select("year", "month", "sumMonth").groupBy("
year", "month").agg(F.avg("sumMonth").alias("avgMonthPrec")).orderBy(["year","month"], as
cending=[False,False])

# Saving the results
focal_readings.rdd.coalesce(1,shuffle=True).saveAsTextFile("BDA/output")
```

Row(year=u'2016', month=u'04', avgMonthPrec=26.900000000000006)
Row(year=u'2016', month=u'03', avgMonthPrec=19.962500000000002)
Row(year=u'2015', month=u'10', avgMonthPrec=2.2625)
Row(year=u'2015', month=u'04', avgMonthPrec=15.3375)
Row(year=u'2014', month=u'11', avgMonthPrec=52.425000000000054)
Row(year=u'2014', month=u'08', avgMonthPrec=90.81249999999997)
Row(year=u'2014', month=u'07', avgMonthPrec=22.987500000000004)
Row(year=u'2014', month=u'06', avgMonthPrec=75.13750000000002)
Row(year=u'2014', month=u'05', avgMonthPrec=58.00000000000001)
Row(year=u'2014', month=u'04', avgMonthPrec=31.762500000000006)
Row(year=u'2012', month=u'10', avgMonthPrec=65.58333333333333)
Row(year=u'2012', month=u'09', avgMonthPrec=72.75)
Row(year=u'2012', month=u'08', avgMonthPrec=68.81666666666665)
Row(year=u'2012', month=u'07', avgMonthPrec=59.06666666666667)
Row(year=u'2011', month=u'06', avgMonthPrec=88.35000000000001)
Row(year=u'2011', month=u'04', avgMonthPrec=14.916666666666666)

Mathias Fredholm, matfr953, 961121-8158
David Gumpert Harryson, davha130, 960302-4937

Row(year=u'2011', month=u'03', avgMonthPrec=19.833333333333336)
Row(year=u'2010', month=u'11', avgMonthPrec=93.54999999999994)
Row(year=u'2010', month=u'09', avgMonthPrec=43.08333333333335)
Row(year=u'2010', month=u'08', avgMonthPrec=108.05)
Row(year=u'2009', month=u'08', avgMonthPrec=61.56666666666667)
Row(year=u'2009', month=u'07', avgMonthPrec=113.16666666666663)
Row(year=u'2009', month=u'06', avgMonthPrec=49.76666666666667)