Big Data Analytics LAB2

Nicolas Taba (nicta839) & Tim Yuki Washio (timwa902)

5/5/2021

Assignment 1

```
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("BDA/input/temperature-readings.csv")

parts = rdd.map(lambda line: line.split(";"))

temp_readings = parts.map(lambda p: Row(
    station=p[0],
    year=p[1].split("-")[0],
    value=float(p[3])))

df_temps = sqlContext.createDataFrame(temp_readings)
df_temps.registerTempTable("temp_readings")

df_temps = df_temps.filter((df_temps["year"] >= 1950) & (df_temps["year"] <= 2014))

max_temps = df_temps.groupBy("year").agg(F.max("value").alias("value"))
max_temps = max_temps.join(df_temps.select(["year", "station", "value"]), ["year", "value"], 'left') \
    .withColumnRenamed("value", "maxValue") \
    .orderBy(["maxValue", "year"], ascending=[0,1])

min_temps = df_temps.groupBy("year").agg(F.min("value").alias("value"))
min_temps = min_temps.join(df_temps.select(["year", "station", "value"]), ["year", "value"], 'left') \
    .withColumnRenamed("value", "minValue") \
    .orderBy(["minValue", "year"], ascending=[0,1])

max_temps.rdd.saveAsTextFile("BDA/output/LAB2_1_MAX")
min_temps.rdd.saveAsTextFile("BDA/output/LAB2_1_MIN")
```

Output

```
# Maximum temperatures
Row(year=u'1975', maxValue=36.1, station=u'86200')
Row(year=u'1992', maxValue=35.4, station=u'63600')
Row(year=u'1994', maxValue=34.7, station=u'117160')
```

```
Row(year=u'2010', maxValue=34.4, station=u'75250')
Row(year=u'2014', maxValue=34.4, station=u'96560')
Row(year=u'1989', maxValue=33.9, station=u'63050')
Row(year=u'1982', maxValue=33.8, station=u'94050')
Row(year=u'1968', maxValue=33.7, station=u'137100')
Row(year=u'1966', maxValue=33.5, station=u'151640')
Row(year=u'1983', maxValue=33.3, station=u'98210')

# Minimum temperatures
Row(year=u'1966', minValue=-49.4, station=u'179950')
Row(year=u'1999', minValue=-49.0, station=u'192830')
Row(year=u'1978', minValue=-47.7, station=u'155940')
Row(year=u'1987', minValue=-47.3, station=u'123480')
Row(year=u'1967', minValue=-45.4, station=u'166870')
Row(year=u'1980', minValue=-45.0, station=u'191900')
Row(year=u'1956', minValue=-45.0, station=u'160790')
Row(year=u'1971', minValue=-44.3, station=u'166870')
Row(year=u'1986', minValue=-44.2, station=u'167860')
Row(year=u'2001', minValue=-44.0, station=u'112530')
```

Assignment 2

```
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("BDA/input/temperature-readings.csv")

parts = rdd.map(lambda line: line.split(";"))

temp_readings = parts.map(lambda p: Row(
    station=p[0],
    year=p[1].split("-")[0],
    month=p[1].split("-")[1],
    value=float(p[3])))

df_temps = sqlContext.createDataFrame(temp_readings)
df_temps.registerTempTable("temp_readings")

df_temps = df_temps.filter((df_temps["year"] >= 1950) & \
    (df_temps["year"] <= 2014) & \
    (df_temps["value"] > 10)) \
    .drop_duplicates()

# Duplicate counts for each station included
df_temps_grouped = df_temps.groupBy(["year", "month"]).count().sort("count", ascending=False)
df_temps_grouped.rdd.saveAsTextFile("BDA/output/LAB2_2")

# Distinct monthly counts
df_temps_distinct = df_temps.select(["year", "month", "station"]).distinct()
```

```
df_temps_grouped_distinct = df_temps_distinct.groupBy(["year", "month"]).count().sort("count", ascending
df_temps_grouped_distinct.rdd.saveAsTextFile("BDA/output/LAB2_2_DISTINCT")
```

Output

```
# Temperature counts above 10 degrees between 1950 and 2014
Row(year=u'2014', month=u'07', count=39041)
Row(year=u'2003', month=u'07', count=34597)
Row(year=u'2008', month=u'07', count=34533)
Row(year=u'2010', month=u'07', count=34397)
Row(year=u'2011', month=u'06', count=34005)
Row(year=u'1997', month=u'08', count=33860)
Row(year=u'2005', month=u'07', count=33779)
Row(year=u'2006', month=u'07', count=33581)
Row(year=u'1997', month=u'07', count=32923)
Row(year=u'2002', month=u'08', count=32413)

# Distinct temperature counts above 10 degrees between 1950 and 2014
Row(year=u'1972', month=u'10', count=378)
Row(year=u'1973', month=u'06', count=377)
Row(year=u'1973', month=u'05', count=377)
Row(year=u'1973', month=u'09', count=376)
Row(year=u'1972', month=u'08', count=376)
Row(year=u'1972', month=u'05', count=375)
Row(year=u'1972', month=u'09', count=375)
Row(year=u'1972', month=u'06', count=375)
Row(year=u'1971', month=u'08', count=375)
Row(year=u'1971', month=u'09', count=374)
```

Assignment 3

```
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("BDA/input/temperature-readings.csv")

parts = rdd.map(lambda line: line.split(";"))

temp_readings = parts.map(lambda p: Row(
    station=p[0],
    year=p[1].split("-")[0],
    month=p[1].split("-")[1],
    day=p[1].split("-")[2],
    value=float(p[3])))

df_temps = sqlContext.createDataFrame(temp_readings)
df_temps.registerTempTable("temp_readings")
```

```
# Filter entries
df_temps = df_temps.filter((df_temps["year"] >= 1960) & (df_temps["year"] <= 2014))

# Calculate Max and Min temperatures
df_temps = df_temps.groupBy("station", "year", "month", "day") \
    .agg(F.max("value").alias("maxTemp"), F.min("value").alias("minTemp"))

# Create new column calculating the average from Max and Min temperature column
df_temps = df_temps.withColumn("avgTemp", (df_temps["maxTemp"] + df_temps["minTemp"]) / 2)

# Grouping on month and year and calculating the average monthly temperature
df_temps = df_temps.groupBy(["station", "month", "year"]) \
    .agg(F.avg("avgTemp").alias("monthlyAvgTemp")) \
    .orderBy(["monthlyAvgTemp", "year"], ascending=[0,1])

df_temps.rdd.saveAsTextFile("BDA/output/LAB2_3")
```

Output

```
# Average monthly temperature for each station between 1960 and 2014
Row(station=u'96000', month=u'07', year=u'2014', monthlyAvgTemp=26.3)
Row(station=u'96550', month=u'07', year=u'1994', monthlyAvgTemp=23.071052631578947)
Row(station=u'54550', month=u'08', year=u'1983', monthlyAvgTemp=23.0)
Row(station=u'78140', month=u'07', year=u'1994', monthlyAvgTemp=22.97096774193548)
Row(station=u'85280', month=u'07', year=u'1994', monthlyAvgTemp=22.872580645161293)
Row(station=u'75120', month=u'07', year=u'1994', monthlyAvgTemp=22.858064516129033)
Row(station=u'65450', month=u'07', year=u'1994', monthlyAvgTemp=22.856451612903232)
Row(station=u'96000', month=u'07', year=u'1994', monthlyAvgTemp=22.808064516129033)
Row(station=u'95160', month=u'07', year=u'1994', monthlyAvgTemp=22.764516129032252)
Row(station=u'86200', month=u'07', year=u'1994', monthlyAvgTemp=22.711290322580645)
Row(station=u'78140', month=u'08', year=u'2002', monthlyAvgTemp=22.7)
```

Assignment 4

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

sc = SparkContext()
sqlContext = SQLContext(sc)

# Temperature data setup
rdd_temp = sc.textFile("BDA/input/temperature-readings.csv")
parts_temp = rdd_temp.map(lambda line: line.split(";"))

temp_readings = parts_temp.map(lambda p: Row( \
    station=p[0], \
    temperature=float(p[3])))

df_temps = sqlContext.createDataFrame(temp_readings)
df_temps.registerTempTable("temp_readings")

# Precipitation data setup
```

```
rdd_precip = sc.textFile("BDA/input/precipitation-readings.csv")
# rdd_precip = sc.textFile("BDA_demo/input_data/precipitation-readings.csv")
parts_precip = rdd_precip.map(lambda line: line.split(";"))

precip_readings = parts_precip.map(lambda p: Row(
    day=p[1].split("-")[2],
    month=p[1].split("-")[1],
    year=p[1].split("-")[0],
    station=p[0],
    precipitation=float(p[3])))

df_precip = sqlContext.createDataFrame(precip_readings)
df_precip.registerTempTable("precip_readings")

# Summing up hourly precipitation values
df_precip = df_precip.groupBy(["station", "day", "month", "year"]) \
    .agg(F.sum("precipitation") \
    .alias("precipitation"))

# Get maximum temperature and precipitation by station
max_temps = df_temps.groupBy("station") \
    .agg(F.max("temperature").alias("maxTemperature"))

max_precips = df_precip.groupBy("station") \
    .agg(F.max("precipitation").alias("maxPrecipitation"))

# Filter entries with temperature between 25 and 30 degrees
max_temps = max_temps.filter( \
    (max_temps["maxTemperature"] > 25) & \
    (max_temps["maxTemperature"] < 30))

# Filter entries with precipitation between 100mm and 200mm
max_precips = max_precips.filter( \
    (max_precips["maxPrecipitation"] > 100) & \
    (max_precips["maxPrecipitation"] < 200))

# Join dataframes keep only those entries that have data for both maxTemperature and maxPrecipitation
df_output = max_temps.join(max_precips, "station", 'inner') \
    .sort("station") \
    .select(["station", "maxTemperature", "maxPrecipitation"])

df_output.rdd.saveAsTextFile("BDA/output/LAB2_4")
```

Output

```
# Maximum measured temperature and precipitation for each station where
# 25 degrees <= temperature <= 30 degrees and
# 100mm <= precipitation <= 200mm

# 0 stations found!
```

Assignment 5

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

sc = SparkContext()
sqlContext = SQLContext(sc)

# Precipitation
rdd_precip = sc.textFile("BDA/input/precipitation-readings.csv")
parts_precip = rdd_precip.map(lambda line: line.split(";"))

precip_readings = parts_precip.map(lambda p: Row(
    station=p[0],
    month=p[1].split("-")[1],
    year=p[1].split("-")[0],
    precipitation=float(p[3])))

df_precip = sqlContext.createDataFrame(precip_readings)
df_precip.registerTempTable("precip_readings")

# Stations
rdd_stations = sc.textFile("BDA/input/stations-Ostergotland.csv")
parts_stations = rdd_stations.map(lambda line: line.split(";"))

# Filter data by year requirements (1993-2016)
df_precip = df_precip.filter((df_precip["year"] >= 1993) & (df_precip["year"] <= 2016))

# Collecting RDD and broadcasting the local python object to all nodes
station_ids_oster = parts_stations.map(lambda x: x[0])
stations_ids = sc.broadcast(station_ids_oster.collect()).value

# Keep only those stations inside Ostergotland
df_precip = df_precip[df_precip["station"].isin(stations_ids)] \
    .sort(["year", "month"]) \
    .select(["station", "year", "month", "precipitation"])

# Sum up precipitation per station per month and year
df_sum = df_precip.groupBy(["station", "year", "month"]) \
    .agg(F.sum("precipitation") \
    .alias("totalPrecipitation"))

# Avg over all stations
df_avg = df_sum.groupBy("year", "month") \
    .agg(F.avg("totalPrecipitation") \
    .alias("avgPrecipitation")) \
    .sort(["year", "month"])

df_avg.rdd.saveAsTextFile("BDA/output/LAB2_5")
```

Output

```
# Average monthly precipitation for each month, year and station in Ostergotland

Row(year=u'1993', month=u'04', avgPrecipitation=0.0)
Row(year=u'1993', month=u'05', avgPrecipitation=21.100000000000005)
Row(year=u'1993', month=u'06', avgPrecipitation=56.5)
Row(year=u'1993', month=u'07', avgPrecipitation=95.39999999999999)
Row(year=u'1993', month=u'08', avgPrecipitation=80.70000000000005)
Row(year=u'1993', month=u'09', avgPrecipitation=40.6)
Row(year=u'1993', month=u'10', avgPrecipitation=43.2)
Row(year=u'1993', month=u'11', avgPrecipitation=42.80000000000003)
Row(year=u'1993', month=u'12', avgPrecipitation=37.10000000000003)
```