Thijs Quast (thiqu264)

Alexander Karlsson (aleka769)

BDA2 - sparkSQL

732A54 – Big Data Analytics

6 may 2019

# Question 1

Highest and lowest temperature measurements between 1950-2014:

Row(year=1975, station=u'86200', value=36.1)

Row(year=1992, station=u'63600', value=35.4)

Row(year=1994, station=u'117160', value=34.7)

Row(year=2010, station=u'75250', value=34.4)

Row(year=2014, station=u'96560', value=34.4)

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

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

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

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

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

## Question 2

**Spark SQL:**

Number of readings:

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)

Number of distinct readings:

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)

**Regular SQL:**

Number of readings:

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)

Number of distinct readings:

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

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

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

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

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

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

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

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

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

## Question 3

Row(year=2014, month=7, station=96000, avg\_temp=26.3)

Row(year=1994, month=7, station=96550, avg\_temp=23.071052631578947)

Row(year=1983, month=8, station=54550, avg\_temp=23.0)

Row(year=1994, month=7, station=78140, avg\_temp=22.970967741935485)

Row(year=1994, month=7, station=85280, avg\_temp=22.872580645161293)

Row(year=1994, month=7, station=75120, avg\_temp=22.858064516129033)

Row(year=1994, month=7, station=65450, avg\_temp=22.85645161290323)

Row(year=1994, month=7, station=96000, avg\_temp=22.80806451612903)

Row(year=1994, month=7, station=95160, avg\_temp=22.764516129032263)

Row(year=1994, month=7, station=86200, avg\_temp=22.71129032258064)

## Question 4

[x\_thiqu@heffa1 Desktop]$ hdfs dfs -cat q4\_sql/part-00000

[x\_thiqu@heffa1 Desktop]$

## Question 5

((2016, 7), 0.0)

((2016, 6), 47.662499999999994)

((2016, 5), 29.25)

((2016, 4), 26.9)

((2016, 3), 19.9625)

((2016, 2), 21.5625)

((2016, 1), 22.325)

((2015, 12), 28.925)

((2015, 11), 63.8875)

## Question 6

((2014, 12), 1.0535912750100962)

((2014, 11), 2.1448898517309694)

((2014, 10), 1.548075577831339)

((2014, 9), 0.04513361330729282)

((2014, 8), -0.7808488781635319)

((2014, 7), 2.147614024320493)

((2014, 6), -2.116175094839768)

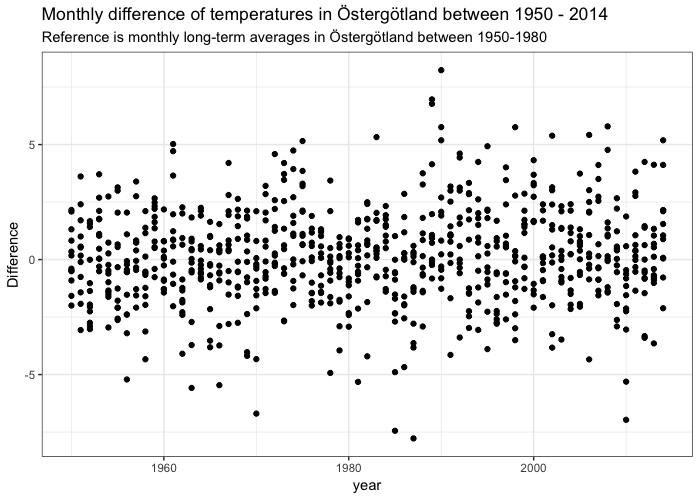
((2014, 5), 0.0761415441317439)

((2014, 4), 2.1010332098100495)

((2014, 3), 4.115318666491132)

((2014, 2), 5.1866962443613005)

((2014, 1), 0.889426352314437)



# Appendix

## Question 1

import pyspark

from pyspark import SparkContext

from pyspark.sql import SQLContext, Row

from pyspark.sql import functions as F

sc = pyspark.SparkContext(appName="Q1\_")

sqlContext = SQLContext(sc)

temp\_path = "/user/x\_thiqu/data/temperature-readings.csv"

temperature\_file = sc.textFile(temp\_path)

parts = temperature\_file.map(lambda l: l.split(";"))

tempReadingsRow = parts.map(lambda p: (p[0], p[1], int(p[1].split("-")[0]),

int(p[1].split("-")[1]), p[2], float(p[3]), p[4]))

tempReadingsString = ["station", "date", "year", "month", "time", "value",

"quality"]

schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)

schemaTempReadings.registerTempTable("tempReadingsTable")

schemaTempReadingsMax = schemaTempReadings.select('year', 'station', 'value')

schemaTempReadingsMax = schemaTempReadings.groupBy(

'year').agg(F.max(

'value').alias('value')).orderBy([

'value'], ascending=[0,0,0,1])

joined = schemaTempReadingsMax.join(schemaTempReadings, ["year", "value"])

joined = joined.select("year", "station", "value")

joined = joined.orderBy('value', ascending=[0,0,1])

joined = joined.where(joined['year'] >= 1950)

joined=joined.where(joined['year'] <= 2014)

schemaTempReadingsMax = joined.rdd

schemaTempReadingsMax = schemaTempReadingsMax.sortBy(ascending=False, keyfunc = lambda k: k[2], numPartitions=1)

schemaTempReadingsMax.saveAsTextFile("q1\_sql\_1")

import pyspark

from pyspark import SparkContext

from pyspark.sql import SQLContext, Row

from pyspark.sql import functions as F

sc = pyspark.SparkContext(appName="Q1\_")

sqlContext = SQLContext(sc)

temp\_path = "/user/x\_thiqu/data/temperature-readings.csv"

temperature\_file = sc.textFile(temp\_path)

parts = temperature\_file.map(lambda l: l.split(";"))

tempReadingsRow = parts.map(lambda p: (p[0], p[1], int(p[1].split("-")[0]),

int(p[1].split("-")[1]), p[2], float(p[3]), p[4]))

tempReadingsString = ["station", "date", "year", "month", "time", "value",

"quality"]

schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)

schemaTempReadings.registerTempTable("tempReadingsTable")

schemaTempReadingsMax = schemaTempReadings.select('year', 'station', 'value')

schemaTempReadingsMax = schemaTempReadings.groupBy(

'year').agg(F.min(

'value').alias('value')).orderBy([

'value'], ascending=[0,0,0,1])

joined = schemaTempReadingsMax.join(schemaTempReadings, ["year", "value"])

joined = joined.select("year", "station", "value")

joined = joined.orderBy('value', ascending=[0,0,1])

joined = joined.where(joined['year'] >= 1950)

joined=joined.where(joined['year'] <= 2014)

schemaTempReadingsMax = joined.rdd

schemaTempReadingsMax = schemaTempReadingsMax.sortBy(ascending=False, keyfunc = lambda k: k[2], numPartitions=1)

schemaTempReadingsMax.saveAsTextFile("q1\_sql\_2")

## Question 2

**API, number of distinct readings part 1**

import pyspark

from pyspark import SparkContext

from pyspark.sql import SQLContext, Row

from pyspark.sql import functions as F

sc = pyspark.SparkContext(appName="Q1\_")

sqlContext = SQLContext(sc)

temp\_path = "/user/x\_thiqu/data/temperature-readings.csv"

temperature\_file = sc.textFile(temp\_path)

parts = temperature\_file.map(lambda l: l.split(";"))

tempReadingsRow = parts.map(lambda p: (int(p[0]), p[1], int(p[1].split("-")[0]),

int(p[1].split("-")[1]), p[2], float(p[3]), p[4], 1))

tempReadingsString = ["station", "date", "year", "month", "time", "value",

"quality", "count"]

schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)

schemaTempReadings.registerTempTable("tempReadingsTable")

schemaTempReadingsMax = schemaTempReadings.select('year', 'month', 'value', 'count')

schemaTempReadingsMax = schemaTempReadingsMax.where(schemaTempReadingsMax['value'] > 10)

schemaTempReadingsMax = schemaTempReadingsMax.where(schemaTempReadingsMax['year'] >= 1950)

schemaTempReadingsMax = schemaTempReadingsMax.where(schemaTempReadingsMax['year'] <= 2014)

schemaTempReadingsMax = schemaTempReadingsMax.select("year", "month", "count")

schemaTempReadingsMax = schemaTempReadingsMax.groupBy(

'year', 'month').agg(F.sum(

'count').alias('count')).orderBy([

'count'], ascending=[0,0,1])

schemaTempReadingsMax = schemaTempReadingsMax.rdd

schemaTempReadingsMax = schemaTempReadingsMax.sortBy(ascending=False, keyfunc = lambda k: k[2], numPartitions=1)

schemaTempReadingsMax.saveAsTextFile("q2\_sql\_1")

**API, number of distinct readings part 2**

import pyspark

from pyspark import SparkContext

from pyspark.sql import SQLContext, Row

from pyspark.sql import functions as F

sc = pyspark.SparkContext(appName="Q1\_")

sqlContext = SQLContext(sc)

temp\_path = "/user/x\_thiqu/data/temperature-readings.csv"

temperature\_file = sc.textFile(temp\_path)

parts = temperature\_file.map(lambda l: l.split(";"))

tempReadingsRow = parts.map(lambda p: (p[0], p[1], int(p[1].split("-")[0]),

int(p[1].split("-")[1]), p[2], float(p[3]), p[4]))

tempReadingsString = ["station", "date", "year", "month", "time", "value",

"quality"]

schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)

schemaTempReadings.registerTempTable("tempReadingsTable")

schemaTempReadingsMax = schemaTempReadings.select('year', 'station', 'value')

schemaTempReadingsMax = schemaTempReadings.groupBy(

'year').agg(F.min(

'value').alias('value')).orderBy([

'value'], ascending=[0,0,0,1])

joined = schemaTempReadingsMax.join(schemaTempReadings, ["year", "value"])

joined = joined.select("year", "station", "value")

joined = joined.orderBy('value', ascending=[0,0,1])

joined = joined.where(joined['year'] >= 1950)

joined=joined.where(joined['year'] <= 2014)

schemaTempReadingsMax = joined.rdd

schemaTempReadingsMax = schemaTempReadingsMax.sortBy(ascending=False, keyfunc = lambda k: k[2], numPartitions=1)

schemaTempReadingsMax.saveAsTextFile("q1\_sql\_2")

**Non-API SQL, number of distinct readings part 1**

import pyspark

from pyspark import SparkContext

from pyspark.sql import SQLContext, Row

from pyspark.sql import functions as F

sc = pyspark.SparkContext(appName="Q1\_")

sqlContext = SQLContext(sc)

temp\_path = "/user/x\_thiqu/data/temperature-readings.csv"

temperature\_file = sc.textFile(temp\_path)

parts = temperature\_file.map(lambda l: l.split(";"))

tempReadingsRow = parts.map(lambda p: (int(p[0]), p[1], int(p[1].split("-")[0]),

int(p[1].split("-")[1]), p[2], float(p[3]), p[4]))

tempReadingsString = ["station", "date", "year", "month", "time", "value",

"quality"]

schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)

schemaTempReadings.registerTempTable("tempReadingsTable")

largerThan10Degrees = sqlContext.sql("SELECT year, month, count(value) as count FROM tempReadingsTable WHERE year >= 1950 and year <= 2014 and value>10.0 group by year, month order by count DESC")

largerThan10Degrees = largerThan10Degrees.rdd

largerThan10Degrees = largerThan10Degrees.sortBy(ascending=False, keyfunc = lambda k: k[2], numPartitions=1)

largerThan10Degrees.saveAsTextFile("q2\_sql\_SQL\_1")

**Non-API SQL, number of distinct readings part 2**

import pyspark

from pyspark import SparkContext

from pyspark.sql import SQLContext, Row

from pyspark.sql import functions as F

sc = pyspark.SparkContext(appName="Q1\_")

sqlContext = SQLContext(sc)

temp\_path = "/user/x\_alkar/data/temperature-readings.csv"

temperature\_file = sc.textFile(temp\_path)

parts = temperature\_file.map(lambda l: l.split(";"))

tempReadingsRow = parts.map(lambda p: (int(p[0]), p[1], int(p[1].split("-")[0]),

int(p[1].split("-")[1]), p[2], float(p[3]), p[4]))

tempReadingsString = ["station", "date", "year", "month", "time", "value",

"quality"]

schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)

schemaTempReadings.registerTempTable("tempReadingsTable")

max1950 = sqlContext.sql("SELECT year, month, station, value FROM tempReadingsTable WHERE year >= 1950 and year <= 2014")

max1950.registerTempTable("temp1950")

max1950 = sqlContext.sql("SELECT year, month, station, value FROM temp1950 WHERE value > 10")

max1950.registerTempTable("temp1950")

max1950 = sqlContext.sql("SELECT year, month, station, max(value) as max FROM temp1950 group by year, month, station")

max1950.registerTempTable("temp1950")

max1950 = sqlContext.sql("SELECT year, month, count(max) as count FROM temp1950 group by year, month order by count DESC")

max1950 = max1950.rdd

max1950 = max1950.sortBy(ascending=False, keyfunc = lambda k: k[2], numPartitions=1)

max1950.saveAsTextFile("q2\_sql\_SQL\_2")

## Question 3

import pyspark

from pyspark import SparkContext

from pyspark.sql import SQLContext, Row

from pyspark.sql import functions as F

sc = pyspark.SparkContext(appName="Q1\_")

sqlContext = SQLContext(sc)

temp\_path = "/user/x\_thiqu/data/temperature-readings.csv"

temperature\_file = sc.textFile(temp\_path)

parts = temperature\_file.map(lambda l: l.split(";"))

tempReadingsRow = parts.map(lambda p: (int(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"]

schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)

schemaTempReadings.registerTempTable("tempReadingsTable")

schemaTempReadingsMax = schemaTempReadings.select('year', 'station','month',"day", 'value')

schemaTempReadingsMax\_full = schemaTempReadingsMax.where(schemaTempReadingsMax['year'] >= 1960)

schemaTempReadingsMax\_full = schemaTempReadingsMax.where(schemaTempReadingsMax['year'] <= 2014)

schemaTempReadingsMax = schemaTempReadingsMax\_full.groupBy(

'year', 'month', 'station', 'day').agg(F.max(

'value').alias('max')).orderBy([

'max'], ascending=[0,0,0,1])

schemaTempReadingsMax2 = schemaTempReadingsMax\_full.groupBy(

'year', 'month', 'station', 'day').agg(F.min(

'value').alias('min')).orderBy([

'min'], descending=[0,0,0,0])

data\_joined = schemaTempReadingsMax.join(schemaTempReadingsMax2, ["year", "month", "station", "day"])

data\_joined = data\_joined.withColumn("daily\_mean", (F.col("max") + F.col("min"))/2)

data\_joined = data\_joined.select("year", "month", "station", "day", "daily\_mean")

data\_joined = data\_joined.groupBy(

'year', 'month', 'station').agg(F.avg(

'daily\_mean').alias('avg\_temp')).orderBy([

'avg\_temp'], ascending=[0,0,0,1])

data\_joined = data\_joined.rdd

data\_joined = data\_joined.sortBy(ascending=False, keyfunc = lambda k: k[3], numPartitions=1)

data\_joined.saveAsTextFile("q3\_sql")

## Question 4

import pyspark

from pyspark import SparkContext

from pyspark.sql import SQLContext, Row

from pyspark.sql import functions as F

sc = pyspark.SparkContext(appName="Q1\_")

sqlContext = SQLContext(sc)

temp\_path = "/user/x\_thiqu/data/temperature-readings.csv"

temperature\_file = sc.textFile(temp\_path)

prec\_path = "/user/x\_thiqu/data/precipitation-readings.csv"

precipitation\_file = sc.textFile(prec\_path)

parts = temperature\_file.map(lambda l: l.split(";"))

parts2 = precipitation\_file.map(lambda l: l.split(";"))

tempReadingsRow = parts.map(lambda p: (int(p[0]), int(p[1].split("-")[0]),

int(p[1].split("-")[1]), int(p[1].split("-")[2]), p[2], float(p[3]), p[4]))

precReadingsRow = parts2.map(lambda p: (int(p[0]), int(p[1].split("-")[0]),

int(p[1].split("-")[1]), int(p[1].split("-")[2]), float(p[3])))

tempReadingsString = ["station", "year", "month", "day", "time", "value",

"quality"]

precReadingsString = ["station", "year", "month", "day", "precipitation"]

schemaPrecReadings = sqlContext.createDataFrame(precReadingsRow, precReadingsString)

schemaPrecReadings.registerTempTable("precReadingsTable")

schemaPrecReadingsMax = schemaPrecReadings

schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)

schemaTempReadings.registerTempTable("tempReadingsTable")

schemaTempReadingsMax = schemaTempReadings.select('year', 'station','month',"day", 'value')

schemaPrecReadingsMax = schemaPrecReadingsMax.groupBy(

'station', 'year', 'month', 'day').agg(F.sum(

'precipitation').alias('total\_prec')).orderBy([

'station'], ascending=[0,0,0,0,0])

schemaPrecReadingsMax = schemaPrecReadingsMax.groupBy(

'station').agg(F.max(

'total\_prec').alias('max\_prec')).orderBy([

'station'], ascending=[0,0])

schemaTempReadingsMax = schemaTempReadingsMax.groupBy(

'station').agg(F.max(

'value').alias('max\_temp')).orderBy([

'station'], ascending=[0,0])

data\_joined = schemaTempReadingsMax.join(schemaPrecReadingsMax, ["station"])

data\_joined = data\_joined.where(data\_joined["max\_temp"] > 25).where(data\_joined["max\_temp"] < 30)

data\_joined = data\_joined.where(data\_joined["max\_prec"] > 100).where(data\_joined["max\_prec"] < 200)

data\_joined = data\_joined.rdd

data\_joined = data\_joined.sortBy(ascending=False, keyfunc = lambda k: k[0], numPartitions=1)

data\_joined.saveAsTextFile("q4\_sql")

## Question 5

import pyspark

from pyspark import SparkContext

from pyspark.sql import SQLContext, Row

from pyspark.sql import functions as F

sc = pyspark.SparkContext(appName="Q1\_")

sqlContext = SQLContext(sc)

ost\_path = "/user/x\_thiqu/data/stations-Ostergotland.csv"

ostergotland\_file = sc.textFile(ost\_path)

prec\_path = "/user/x\_thiqu/data/precipitation-readings.csv"

precipitation\_file = sc.textFile(prec\_path)

parts = ostergotland\_file.map(lambda l: l.split(";"))

parts2 = precipitation\_file.map(lambda l: l.split(";"))

statOstergotlandRow = parts.map(lambda p: (int(p[0]), p[1]))

precReadingsRow = parts2.map(lambda p: (int(p[0]), int(p[1].split("-")[0]),

int(p[1].split("-")[1]), int(p[1].split("-")[2]), float(p[3])))

precReadingsString = ["station", "year", "month", "day", "precipitation"]

statOstergotlandString= ["station", "town"]

schemaOstergotlandReadings = sqlContext.createDataFrame(statOstergotlandRow, statOstergotlandString)

schemaOstergotlandReadings.registerTempTable("OstergotlandTable")

schemaPrecReadings = sqlContext.createDataFrame(precReadingsRow, precReadingsString)

schemaPrecReadings.registerTempTable("precReadingsTable")

schemaPrecReadingsMax = schemaPrecReadings

schemaPrecReadingsMax = schemaPrecReadingsMax.groupBy(

'station', 'year', 'month', 'day').agg(F.sum(

'precipitation').alias('total\_prec')).orderBy([

'station'], ascending=[0,0,0,0,0])

schemaPrecReadingsMax = schemaPrecReadingsMax.groupBy(

'station', 'year', 'month').agg(F.sum(

'total\_prec').alias('total\_prec')).orderBy([

'year'], ascending=[0,0,0,0])

data\_joined = schemaOstergotlandReadings.join(schemaPrecReadingsMax, ["station"])

data\_joined = data\_joined.select("year", "month", "total\_prec")

data\_joined = data\_joined.groupBy(

"year", "month").agg(F.avg(

'total\_prec').alias('monthly\_average')).orderBy([

"year"], ascending=[0,0,0])

data\_joined = data\_joined.where(data\_joined["year"] >= 1993).where(data\_joined["year"] <= 2016)

data\_joined = data\_joined.rdd

data\_joined = data\_joined.map(lambda x: ((x[0],x[1]),(x[2])))

data\_joined = data\_joined.sortBy(ascending=False, keyfunc = lambda k: k[0], numPartitions=1)

data\_joined.saveAsTextFile("q5\_sql")

## Question 6

import pyspark

from pyspark import SparkContext

from pyspark.sql import SQLContext, Row

from pyspark.sql import functions as F

sc = pyspark.SparkContext(appName="Q6\_")

sqlContext = SQLContext(sc)

ost\_path = "/user/x\_thiqu/data/stations-Ostergotland.csv"

ostergotland\_file = sc.textFile(ost\_path)

temp\_path = "/user/x\_thiqu/data/temperature-readings.csv"

temperature\_file = sc.textFile(temp\_path)

parts = ostergotland\_file.map(lambda l: l.split(";"))

parts2 = temperature\_file.map(lambda l: l.split(";"))

statOstergotlandRow = parts.map(lambda p: (int(p[0]), p[1]))

tempReadingsRow = parts2.map(lambda p: (int(p[0]), int(p[1].split("-")[0]),

int(p[1].split("-")[1]), int(p[1].split("-")[2]), float(p[3])))

tempReadingsString = ["station", "year", "month", "day", "value"]

schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)

schemaTempReadings.registerTempTable("tempReadingsTable")

schemaTempReadingsMax = schemaTempReadings.select('year', 'month', 'station', 'value')

schemaTempReadingsMax1980 = schemaTempReadingsMax

schemaTempReadingsMax2014 = schemaTempReadingsMax

schemaTempReadingsMax1980 = schemaTempReadingsMax1980.where(schemaTempReadingsMax1980["year"] >= 1950)

schemaTempReadingsMax1980 = schemaTempReadingsMax1980.where(schemaTempReadingsMax1980["year"] <= 1980)

schemaTempReadingsMax2014 = schemaTempReadingsMax2014.where(schemaTempReadingsMax2014["year"] >= 1950)

schemaTempReadingsMax2014 = schemaTempReadingsMax2014.where(schemaTempReadingsMax2014["year"] <= 2014)

statOstergotlandString= ["station", "town"]

schemaOstergotlandReadings = sqlContext.createDataFrame(statOstergotlandRow, statOstergotlandString)

schemaOstergotlandReadings.registerTempTable("OstergotlandTable")

schemaOstergotlandReadings = schemaOstergotlandReadings.select("station")

schemaTempReadingsMax1980 = schemaTempReadingsMax1980.groupBy(

'year', 'month', 'station').agg(F.avg(

'value').alias('avg\_temp')).orderBy([

'year'], ascending=[0,0,0,0])

schemaTempReadingsMax2014 = schemaTempReadingsMax2014.groupBy(

'year', 'month', 'station').agg(F.avg(

'value').alias('avg\_temp')).orderBy([

'year'], ascending=[0,0,0,0])

schemaTempReadingsMax1980 = schemaTempReadingsMax1980.join(schemaOstergotlandReadings, "station")

schemaTempReadingsMax2014 = schemaTempReadingsMax2014.join(schemaOstergotlandReadings, "station")

schemaTempReadingsMax1980 = schemaTempReadingsMax1980.groupBy(

'month').agg(F.avg(

'avg\_temp').alias('avg\_temp\_month')).orderBy([

'month'], ascending = [0,0])

schemaTempReadingsMax2014 = schemaTempReadingsMax2014.groupBy(

'year', 'month').agg(F.avg(

'avg\_temp').alias('avg\_2014')).orderBy([

'year'], ascending= [0,0])

data\_joined = schemaTempReadingsMax1980.join(schemaTempReadingsMax2014, "month")

data\_joined = data\_joined.withColumn("difference", (F.col("avg\_2014") - F.col("avg\_temp\_month")))

data\_joined = data\_joined.select("year", "month", "difference")

data\_joined = data\_joined.rdd

data\_joined = data\_joined.map(lambda x: ((x[0],x[1]),(x[2])))

data\_joined = data\_joined.sortBy(ascending=False, keyfunc = lambda k: k[0], numPartitions=1)

data\_joined.saveAsTextFile("q6\_sql")