Assignments

Q1. 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. The output should at least contain the following information (You can also include a Station column so that you may find multiple stations that record the highest (lowest) temperature.)

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

In [ ]:
```
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 = 'exercise 1')
sqlContext = SQLContext(sc)

#sqlContext = HiveContext(sc)

# This path is to the file on hdfs
temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
lines = temperature_file.map(lambda line: line.split(";"))

# (key, value) = (year,temperature)
tempReadingsRow = lines.map(lambda p: (p[0],int(p[1].split("-")[0]),float(p[3]) ))

## Inferring schema and registering the Dataframe as a table
tempReadingsString = ["station","year","value"]

schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow,tempReadingsString)

# Register the DataFrame as a table
schemaTempReadings.registerTempTable("tempReadingsTable")

#filter years 1950-2014
schemaTempReadings = schemaTempReadings.filter((schemaTempReadings["year"]>= 1950) & (schemaTempReadings["year"]<= 2014))
#schemaTempReadings = schemaTempReadings.collect()


schemaTempReadings =  schemaTempReadings.select(['year','station','value'])

schemaTempReadingsMin = schemaTempReadings.groupBy('year','station').agg(F.min('value').alias('min'))
schemaTempReadingsMinYear = schemaTempReadingsMin.groupBy('year').agg(F.min('min').alias('min'))

year_station_mintemp = schemaTempReadingsMin.join(schemaTempReadingsMinYear, ['year', 'min']).select('year', 'station', 'min').ord


schemaTempReadingsMax = schemaTempReadings.groupBy('year','station').agg(F.max('value').alias('max'))

schemaTempReadingsMaxYear = schemaTempReadingsMax.groupBy('year').agg(F.max('max').alias('max'))

year_station_maxtemp = schemaTempReadingsMax.join(schemaTempReadingsMaxYear, ['year', 'max']).select('year', 'station', 'max').ord

year_station_mintemp.rdd.saveAsTextFile("BDA/output/q1_min")
year_station_maxtemp.rdd.saveAsTextFile("BDA/output/q1_max")
```

- output

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

Row(year=1975, station=u'86200', max=36.1)
Row(year=1992, station=u'63600', max=35.4)
Row(year=1994, station=u'117160', max=34.7)
Row(year=2010, station=u'75250', max=34.4)
Row(year=2014, station=u'96560', max=34.4)
Row(year=1989, station=u'63050', max=33.9)
Row(year=1982, station=u'94050', max=33.8)
Row(year=1968, station=u'137100', max=33.7)
Row(year=1966, station=u'151640', max=33.5)
Row(year=2002, station=u'78290', max=33.3)
Row(year=1983, station=u'98210', max=33.3)
Row(year=1970, station=u'103080', max=33.2)
Row(year=1986, station=u'76470', max=33.2)
Row(year=1956, station=u'145340', max=33.0)
Row(year=2000, station=u'62400', max=33.0)
Row(year=1959, station=u'65160', max=32.8)

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

Row(year=1990, station=u'147270', min=-35.0)
Row(year=1990, station=u'166870', min=-35.0)
Row(year=1952, station=u'192830', min=-35.5)
Row(year=1974, station=u'166870', min=-35.6)
Row(year=1974, station=u'179950', min=-35.6)
Row(year=1954, station=u'113410', min=-36.0)
Row(year=1992, station=u'179960', min=-36.1)
Row(year=1975, station=u'157860', min=-37.0)
Row(year=1972, station=u'167860', min=-37.5)
Row(year=2000, station=u'169860', min=-37.6)
Row(year=1995, station=u'182910', min=-37.6)
Row(year=1957, station=u'159970', min=-37.8)
Row(year=1983, station=u'191900', min=-38.2)
Row(year=1989, station=u'166870', min=-38.2)
Row(year=1953, station=u'183760', min=-38.4)
Row(year=2009, station=u'179960', min=-38.5)
Row(year=1993, station=u'191900', min=-39.0)
Row(year=1984, station=u'191900', min=-39.2)
Row(year=1984, station=u'123480', min=-39.2)

Q2 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 itappears only once in the count for that month.
In this exercise you will use the temperature-readings.csv file.

In [ ]:
```
#Q2
from pyspark import SparkContext
from pyspark.sql import SparkSession, SQLContext
from pyspark.sql.types import StructType, StructField, StringType, FloatType
from pyspark.sql import functions as F

sc = SparkContext(appName="exercise 1")
spark = SparkSession(sc)
sqlContext = SQLContext(sc)

temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
lines = temperature_file.map(lambda line: line.split(";"))

year_month_temperature = lines.map(lambda x: (x[1][0:4],x[1][5:7],x[0],float(x[3])))

schema = StructType([
    StructField("year", StringType(), True),
    StructField("month", StringType(), True),
    StructField("station", StringType(), True),
    StructField("value", FloatType(), True)
])

year_month_temperature_df = sqlContext.createDataFrame(year_month_temperature, schema)

filtered_year_month_temperature = year_month_temperature_df.filter((year_month_temperature_df["year"] >= "1950") & (year_month_tem

count_temp = filtered_year_month_temperature.groupBy(["year", "month"]).count()
sort_count = count_temp.sort("count", ascending = False)
sort_count_combine = sort_count.rdd.coalesce(1)
sort_count_combine = sort_count_combine.sortBy(lambda x: x[2], ascending=False)
sort_count_combine.saveAsTextFile("BDA/output/l2_not_distinct")

distinct_temp = filtered_year_month_temperature.select(["year", "month", "station"]).distinct()
count_dist_temp = distinct_temp.groupBy(["year", "month"]).count()
sort_count_dist = count_dist_temp.sort("count", ascending = False)
sort_count_dist = sort_count_dist.rdd.coalesce(1)
sort_count_dist = sort_count_dist.sortBy(lambda x: x[2], ascending=False)
sort_count_dist.saveAsTextFile("BDA/output/l2_distinct")
```

output of l2_not_distinct

Year, month, count

Row(year=u'2014', month=u'07', count=147681)
Row(year=u'2011', month=u'07', count=146656)
Row(year=u'2010', month=u'07', count=143419)
Row(year=u'2012', month=u'07', count=137477)
Row(year=u'2013', month=u'07', count=133657)
Row(year=u'2009', month=u'07', count=133008)
Row(year=u'2011', month=u'08', count=132734)
Row(year=u'2009', month=u'08', count=128349)
Row(year=u'2013', month=u'08', count=128235)
Row(year=u'2003', month=u'07', count=128133)

output of l2_distinct

Year, month, count

Row(year=u'1972', month=u'10', count=378)
Row(year=u'1973', month=u'05', count=377)
Row(year=u'1973', month=u'06', count=377)
Row(year=u'1972', month=u'08', count=376)
Row(year=u'1973', month=u'09', count=376)
Row(year=u'1972', month=u'06', count=375)
Row(year=u'1972', month=u'09', count=375)
Row(year=u'1971', month=u'08', count=375)
Row(year=u'1972', month=u'05', count=375)
Row(year=u'1971', month=u'06', count=374)

Q3.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.

The output should contain the following information:

year, month, station, avgMonthlyTemperature ORDER BY avgMonthlyTemperature DESC

In [ ]:
```
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 = 'exercise 3')
sqlContext = SQLContext(sc)

# This path is to the file on hdfs
temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
lines = temperature_file.map(lambda line: line.split(";"))

# (key, value) = (year,month,date,station,temperature)
tempReadingsRow = lines.map(lambda x: ( x[1][0:4],x[1][5:7],x[1][8:],x[0] , float(x[3]) ) )

## Inferring schema and registering the Dataframe as a table
tempReadingsString = ["year","month","date","station","temperature"]

schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow,tempReadingsString)

# Register the DataFrame as a table
schemaTempReadings.registerTempTable("tempReadingsTable")

#filter years 1950-2014
schemaTempReadings = schemaTempReadings.filter((schemaTempReadings["year"]>= 1960) & (schemaTempReadings["year"]<= 2014))


schemaTempReadings =  schemaTempReadings.select(['year','month','station','temperature'])

schemaTempReadingsMean = schemaTempReadings.groupBy('year','month','station').agg(F.mean('temperature').alias('avg')).orderBy(['av

schemaTempReadingsMean.rdd.saveAsTextFile("BDA/output")
```

- output of Q3

year, month, station, avgMonthlyTemperature ORDER BY avgMonthlyTemperature DESC

Row(year=u'2014', month=u'07', station=u'96000', avg=26.3)
Row(year=u'1994', month=u'07', station=u'65450', avg=23.65483870967742)
Row(year=u'1994', month=u'07', station=u'95160', avg=23.505376344086027)
Row(year=u'1994', month=u'07', station=u'75120', avg=23.26881720430107)
Row(year=u'1994', month=u'07', station=u'105260', avg=23.143820224719107)
Row(year=u'1994', month=u'07', station=u'85280', avg=23.108602150537635)
Row(year=u'1983', month=u'08', station=u'54550', avg=23.0)
Row(year=u'1975', month=u'08', station=u'54550', avg=22.9625)
Row(year=u'1994', month=u'07', station=u'96550', avg=22.957894736842114)
Row(year=u'1994', month=u'07', station=u'96000', avg=22.931182795698923)
Row(year=u'1994', month=u'07', station=u'106070', avg=22.822580645161295)
Row(year=u'1972', month=u'07', station=u'173960', avg=22.776666666666667)
Row(year=u'1994', month=u'07', station=u'54300', avg=22.76021505376344)
Row(year=u'1994', month=u'07', station=u'85210', avg=22.755913978494615)
Row(year=u'2006', month=u'07', station=u'65450', avg=22.74086021505376)
Row(year=u'2006', month=u'07', station=u'75120', avg=22.73010752688173)
Row(year=u'1994', month=u'07', station=u'103080', avg=22.708602150537626)
Row(year=u'1994', month=u'07', station=u'92100', avg=22.698924731182792)
Row(year=u'1994', month=u'07', station=u'94180', avg=22.68172043010753)
Row(year=u'1994', month=u'07', station=u'83230', avg=22.577419354838707)

Q4 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 200mm.

In [ ]:
```
#Q4
from pyspark import SparkContext
from pyspark.sql import SparkSession, SQLContext
from pyspark.sql.types import StructType, StructField, StringType, FloatType
from pyspark.sql import functions as F

sc = SparkContext(appName="exercise 1")
spark = SparkSession(sc)
sqlContext = SQLContext(sc)

temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
temperature_lines = temperature_file.map(lambda line: line.split(";"))
get_temperature = temperature_lines.map(lambda x: (x[0],float(x[3])))
tempschema = StructType([
    StructField("station", StringType(), True),
    StructField("temp", FloatType(), True)
])
temperature_df = sqlContext.createDataFrame(get_temperature, tempschema)
station_max_temp = temperature_df.groupBy("station").agg(F.max("temp").alias('temp'))
filter_temp = station_max_temp.filter((station_max_temp["temp"] >= "25") & (station_max_temp["temp"] <= "30"))

precipitation_file = sc.textFile("BDA/input/precipitation-readings.csv")
precipitation_file = precipitation_file.map(lambda line: line.split(";"))
get_percipitation = precipitation_file.map(lambda x: (x[0],float(x[3])))
precschema = StructType([
    StructField("station", StringType(), True),
    StructField("prec", FloatType(), True)
])
percipitation_df = sqlContext.createDataFrame(get_percipitation, precschema)
station_max_perc = percipitation_df.groupBy("station").agg(F.max("prec").alias('prec'))
filter_perc = station_max_perc.filter((station_max_perc["prec"] >= "100") & (station_max_perc["prec"] <= "200"))

combine_temp_perc = filter_temp.join(filter_perc.alias('perc'), 'station', 'inner')

#output
combine_temp_perc_combine = combine_temp_perc.rdd.coalesce(1)
filter_temp_combine = combine_temp_perc_combine.sortBy(lambda x: x[0], ascending=False)
filter_temp_combine.saveAsTextFile("BDA/output/l2_perc_temp")


#Testoutput
#filter_temp_combine = filter_temp.rdd.coalesce(1)
#filter_temp_combine = filter_temp_combine.sortBy(lambda x: x[0], ascending=False)
#filter_temp_combine.saveAsTextFile("BDA/output/l2_temptest")

#filter_perc_combine = filter_perc.rdd.coalesce(1)
#filter_perc_combine = filter_perc_combine.sortBy(lambda x: x[0], ascending=False)
#filter_perc_combine.saveAsTextFile("BDA/output/l2_perctest")

#combine_temp_perc_combine = combine_temp_perc.rdd.coalesce(1)
#filter_temp_combine = combine_temp_perc_combine.sortBy(lambda x: x[0], ascending=False)
#filter_temp_combine.saveAsTextFile("BDA/output/l2_combtest")
```

output of Q4 is empty, since no data meet the criteria

Q5. Calculate the average monthly precipitation for the Östergotland region

(list of stations is provided in the separate file) for the period 1993-2016. In orderto dothis, you will first need to calculate the total monthly precipitation for each station before calculating the monthly average (by averaging over stations). In this exercise you will use the precipitation-readings.csv and stations-Ostergotland.csv files. HINT (not for the SparkSQL lab): Avoid using joins here! stations-Ostergotland.csv is small and if distributed will cause a number of unnecessary shuffles when joined with precipitationRDD. If you distribute precipitation-readings.csv then either repartition your stations RDD to 1 partition or make use of the collect function to acquire a python list and broadcast function to broadcast the list to all nodes. The output should contain the following information:

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

In [ ]:
```
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 = 'exercise 5')
sqlContext = SQLContext(sc)

precipitaion_file = sc.textFile('BDA/input/precipitation-readings.csv')
stations_file = sc.textFile('BDA/input/stations-Ostergotland.csv')

lines = precipitaion_file.map(lambda line: line.split(';'))
stations = stations_file.map(lambda line: line.split(';'))

# (key, value) = (year,month,station,precipitation)
precipReadingRow = lines.map(lambda x: ( x[1][0:4],x[1][5:7],x[0],float(x[3]) ) )
```

```
# (key,value) = (year,month,station,
stationsReadingRow = stations.map(lambda x: (x[0],x[1],x[2],x[3],x[4],x[5],x[6],x[7]))

stationReadingString = ['station','name','height','latitude','longitude','readingfrom','readingto','Elavtion']

## Inferring schema and registering the Dataframe as a table
precipReadingsString = ["year","month","station","precipitation"]

schemaPrecipReadings = sqlContext.createDataFrame(precipReadingRow,precipReadingsString)
schemaStations = sqlContext.createDataFrame(stationsReadingRow,stationReadingString)

# Register the DataFrame as a table
schemaPrecipReadings.registerTempTable("PrecipReadingsTable")
schemaStations.registerTempTable('StationsTable')

schemaStations = schemaStations.select(['station'])

#filter years 1993-2016
schemaPrecipReadings = schemaPrecipReadings.filter((schemaPrecipReadings["year"]>= 1993) & (schemaPrecipReadings["year"]<= 2016))

#join with station
schemaPrecipReadings = schemaPrecipReadings.join(schemaStations,['station'])

schemaPrecipReadings =  schemaPrecipReadings.select(['year','month','station','precipitation'])

#calculate total monthly precipitation
schemaPrecipReadingsMean = schemaPrecipReadings.groupBy('year','month','station').agg(F.sum('precipitation').alias('total')).order

#average over stations
schemaPrecipReadingsMean = schemaPrecipReadingsMean.groupBy('year','month').agg(F.avg('total').alias('avg')).orderBy(['year','mont

schemaPrecipReadingsMean.rdd.saveAsTextFile("BDA/output")
```

- Output of Q5

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

Row(year=u'2016', month=u'07', avg=0.0)
Row(year=u'2016', month=u'06', avg=47.662499999999994)
Row(year=u'2016', month=u'05', avg=29.250000000000004)
Row(year=u'2016', month=u'04', avg=26.900000000000006)
Row(year=u'2016', month=u'03', avg=19.962500000000002)
Row(year=u'2016', month=u'02', avg=21.562500000000004)
Row(year=u'2016', month=u'01', avg=22.325000000000003)
Row(year=u'2015', month=u'12', avg=28.925000000000004)
Row(year=u'2015', month=u'11', avg=63.88750000000002)
Row(year=u'2015', month=u'10', avg=2.2625)
Row(year=u'2015', month=u'09', avg=101.29999999999998)
Row(year=u'2015', month=u'08', avg=26.987499999999997)
Row(year=u'2015', month=u'07', avg=119.09999999999995)
Row(year=u'2015', month=u'06', avg=78.66250000000001)
Row(year=u'2015', month=u'05', avg=93.225)
Row(year=u'2015', month=u'04', avg=15.3375)
Row(year=u'2015', month=u'03', avg=42.612500000000004)
Row(year=u'2015', month=u'02', avg=24.825)
Row(year=u'2015', month=u'01', avg=59.11250000000003)