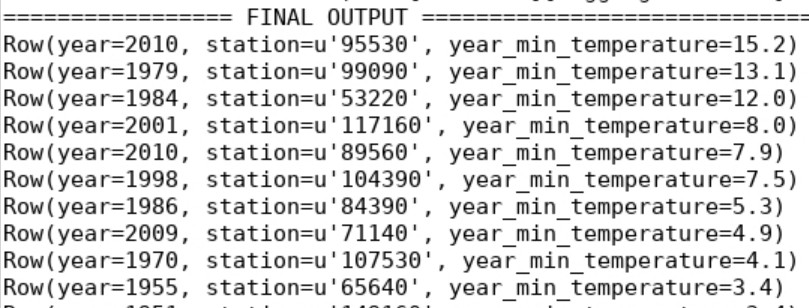
Lab 2 Report

Group member: Hong Zhang(honzh073) Bing Rui Yao(binya518)

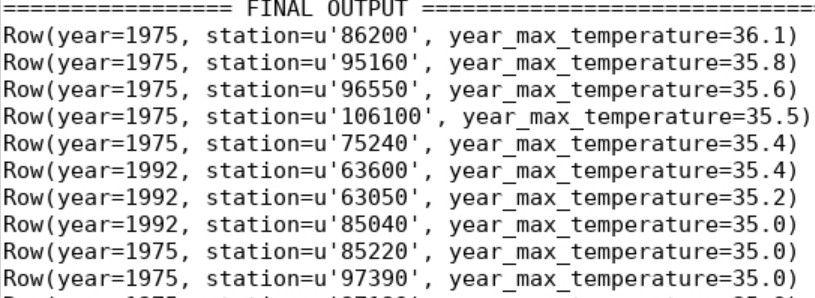
1. What are the lowest and highest temperatures measured each year for the period 1950- 2014. Provide the lists sorted in the descending order with respect to the maximum temperature.

Code:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | from pyspark import SparkContext | | | |
| from pyspark.sql import SQLContext, Row | | | |
| from pyspark.sql import functions as F | | | |
| sc = SparkContext(appName = "exercise 2") | | | |
| 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(";")) | | | |
| tempReadingsRow = lines.map(lambda p: (p[0], p[1], int(p[1].split("-")[0]),int(p[1].split("-")[1]), p[2], | | | |
|  | float(p[3]), p[4] )) | | | | |
|  | | **# Specifying the schema programatically and registering the DataFrame as a table** | | | |
| tempReadingsString = ["station", "date", "year", "month", "time", "value","quality"] | | | |
| **# Apply the schema to the RDD.** | | | |
| tem\_df = sqlContext.createDataFrame(tempReadingsRow,tempReadingsString) | | | |
| # Register the DataFrame as a table. | | | |
| tem\_df.registerTempTable("tempReadingsTable") | | | |
| # Can run queries now | | | |
| #year\_max =tem\_df.filter((tem\_df['year']>=1950) & | |  | |
|  | (tem\_df['year']<=2014)).groupBy('year','station').agg(F.max('value').alias('year\_max\_temperature')).order | | | | |
| By('year\_max\_temperature',ascending=0) | | | | |
| year\_min =tem\_df.filter((tem\_df['year']>=1950) & | |  | | |
| (tem\_df['year']<=2014)).groupBy('year','station').agg(F.min('value').alias('year\_min\_temperature')).orderBy('y | | | | |  |
| ear\_min\_temperature',ascending=0) | | | | | |
|  | | #year\_max.select('year','station','year\_max\_temperature').rdd.saveAsTextFile("BDA/output") | | | |
| year\_min.select('year','station','year\_min\_temperature').rdd.saveAsTextFile("BDA/output") | | | |

First 10 rows of the lowest temperatures:

First 10 rows of the highest temperatures:

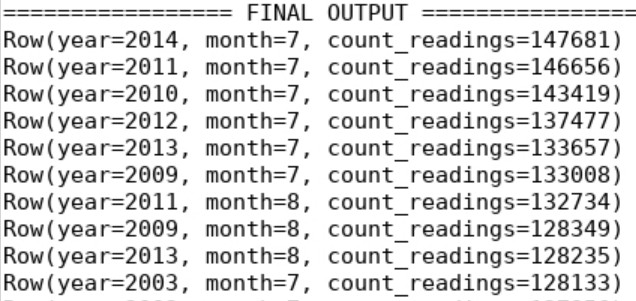


1. Count the number of readings for each month in the period of 1950-2014 which are higher than10degrees.

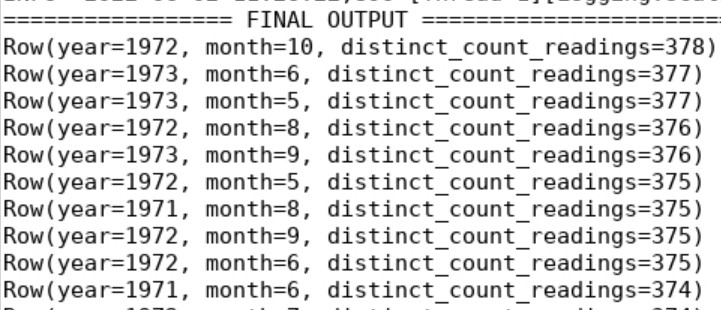
Code:

|  |  |  |
| --- | --- | --- |
|  | from pyspark import SparkContext | |
| from pyspark.sql import SQLContext, Row | |
| from pyspark.sql import functions as F | |
| sc = SparkContext(appName = "exercise 2") | |
| 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(";")) | |
| tempReadingsRow = lines.map(lambda p: (p[0], p[1], int(p[1].split("-")[0]),int(p[1].split("-")[1]), p[2], | |
| float(p[3]), p[4] )) | | |
|  | # Specifying the schema programatically and registering the DataFrame as a table | |
| tempReadingsString = ["station", "date", "year", "month", "time", "value","quality"] | |
| # Apply the schema to the RDD. | |
| tem\_df = sqlContext.createDataFrame(tempReadingsRow,tempReadingsString) | |
| # Register the DataFrame as a table. | |
| tem\_df.registerTempTable("tempReadingsTable") | |
| **#count readings** | |
| #count\_readings =tem\_df.filter((tem\_df['year']>=1950) & (tem\_df['year']<=2014) & |  |
| (tem\_df['value']>10)).groupBy('year','month').agg(F.count('value').alias('count\_readings')).orderBy('count | | |
| \_readings',ascending=0) | | |
|  | **# count dictinct readings** | |
| distinct\_count\_readings = tem\_df.filter((tem\_df['year']>=1950) & (tem\_df['year']<=2014) & | |
| (tem\_df['value']>10)).groupBy('year','month').agg(F.countDistinct('station').alias('distinct\_count\_readings | | |
| ')).orderBy('distinct\_count\_readings',ascending=0) | | |
|  | #count\_readings.rdd.saveAsTextFile("BDA/output") | |
| distinct\_count\_readings.rdd.saveAsTextFile("BDA/output") | |

First 10 rows of output:



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.

First 10 rows of output:

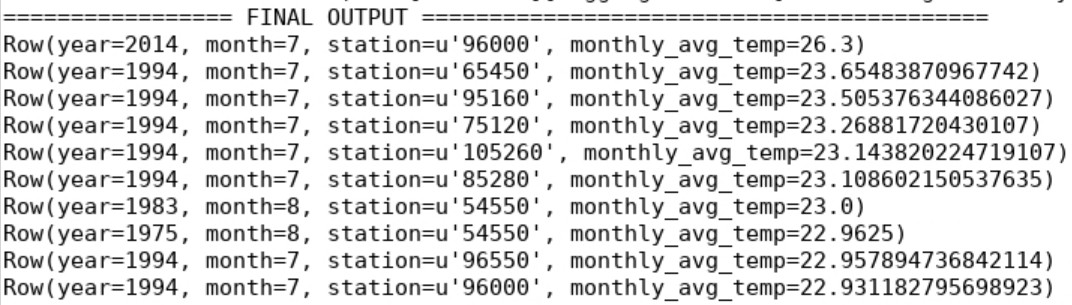
1. Find the average monthly temperature for each available station in Sweden. Your result should include average temperature for each station for each month in the period of 1960- 2014.

Code:

|  |  |
| --- | --- |
|  | from pyspark import SparkContext |
| from pyspark.sql import SQLContext, Row |
| from pyspark.sql import functions as F |
| sc = SparkContext(appName = "exercise 2") |
| 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(";")) |
| tempReadingsRow = lines.map(lambda p: (p[0], p[1], int(p[1].split("-")[0]),int(p[1].split("-")[1]), p[2], |
| float(p[3]), p[4] )) | |
|  | # Specifying the schema programatically and registering the DataFrame as a table |
| tempReadingsString = ["station", "date", "year", "month", "time", "value","quality"] |
| # Apply the schema to the RDD. |
| tem\_df = sqlContext.createDataFrame(tempReadingsRow,tempReadingsString) |
| # Register the DataFrame as a table. |
| tem\_df.registerTempTable("tempReadingsTable") |
| # Can run queries now |

|  |  |  |
| --- | --- | --- |
|  | avg\_monthly = tem\_df.filter((tem\_df['year']>=1960) & |  |
| (tem\_df['year']<=2014)).groupBy('year','month','station').agg(F.avg('value').alias('monthly\_avg\_temp')).o | | |
| rderBy('monthly\_avg\_temp',ascending=0) | | |
|  | avg\_monthly.rdd.saveAsTextFile("BDA/output") | |

First 10 rows of output:



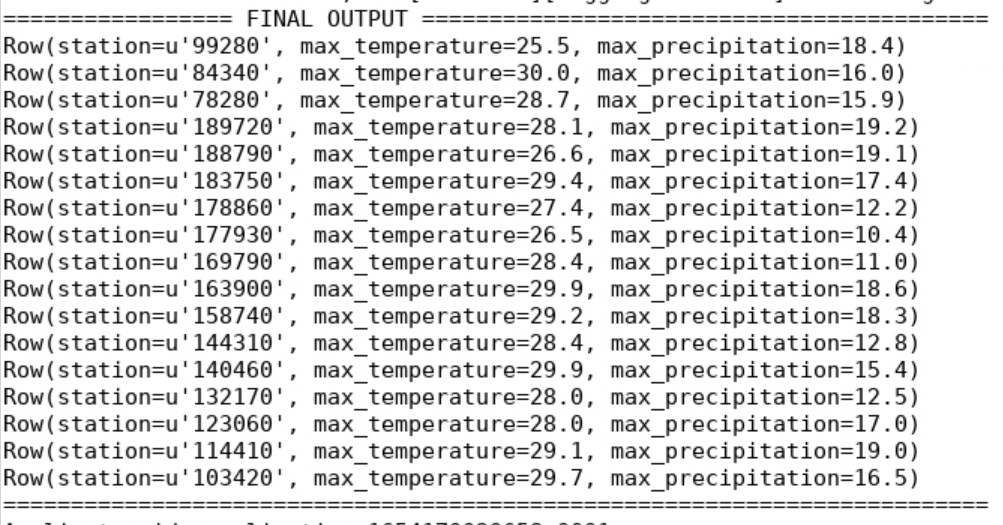
1. Provide a list of stations with their associated maximum measured temperatures and maximum measured daily precipitation. Show only those stations where the maximum temperature is between 25 and 30 degrees and maximum daily precipitation is between 100 mm and 200mm.

Code:

|  |  |  |
| --- | --- | --- |
|  | from pyspark import SparkContext | |
| from pyspark.sql import SQLContext, Row | |
| from pyspark.sql import functions as F | |
| sc = SparkContext(appName = "exercise 2") | |
| sqlContext = SQLContext(sc) | |
| # This path is to the file on hdfs | |
| temperature\_file = sc.textFile("BDA/input/temperature-readings.csv") | |
| precipitation\_file = sc.textFile("BDA/input/precipitation-readings.csv") | |
| temp\_lines = temperature\_file.map(lambda line: line.split(";")) | |
| pre\_lines = precipitation\_file.map(lambda line: line.split(";")) | |
| tempReadingsRow = temp\_lines.map(lambda p: (p[0], p[1], int(p[1].split("-")[0]),int(p[1].split("-")[1]), | |
| p[2], float(p[3]), p[4] )) | | |
|  | preReadingsRow = pre\_lines.map(lambda p: (p[0], p[1], int(p[1].split("-")[0]),int(p[1].split("-")[1]), | |
| p[2], float(p[3]), p[4] )) | | |
|  | # Specifying the schema programatically and registering the DataFrame as a table | |
| tempReadingsString = ["station", "date", "year", "month", "time", "temp\_value","quality"] | |
| preReadingsString = ["station", "date", "year", "month", "time", "pre\_value","quality"] | |
| # Apply the schema to the RDD. | |
| tem\_df = sqlContext.createDataFrame(tempReadingsRow,tempReadingsString) | |
| pre\_df = sqlContext.createDataFrame(preReadingsRow,preReadingsString) | |
| # Register the DataFrame as a table. | |
| tem\_df.registerTempTable("tempReadingsTable") | |
| pre\_df.registerTempTable("preReadingsTable") | |
| # Can run queries now | |
| stations\_temp = tem\_df.groupBy('station').agg(F.max('temp\_value').alias('max\_temperature')) | |
| stations\_temp = stations\_temp.filter((stations\_temp['max\_temperature']>=25) & |  |
| (stations\_temp['max\_temperature']<=30)) | | |
|  | stations\_pre = pre\_df.groupBy('station').agg(F.max('pre\_value').alias('max\_precipitation')) | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | stations\_pre = stations\_pre.filter((stations\_pre['max\_precipitation']>=10) & | |  |
| (stations\_pre['max\_precipitation']<=20)) | | | |
|  | station\_list = |  | |
| stations\_temp.join(stations\_pre,stations\_temp['station']==stations\_pre['station'],'inner').drop(stations\_pr | | | |
| e['station']).orderBy('station',ascending=0) | | | |
| station\_list.rdd.saveAsTextFile("BDA/output") | | | |

First 10 rows of output:

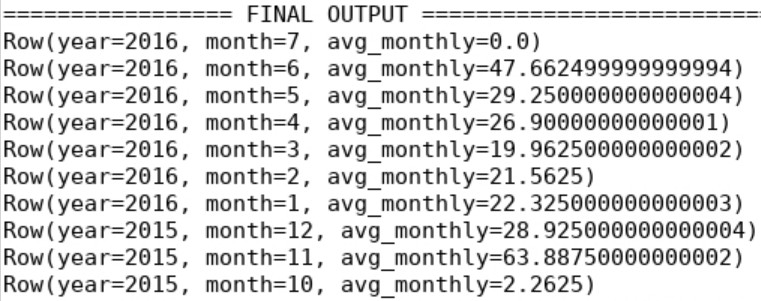


1. Calculate the average monthly precipitation for the Östergotland region (list of stations is provided in the separate file) for the period 1993-2016. In order to do this, you will first need to calculate the total monthly precipitation for each station before calculating the monthly average (by averaging over stations).

Code:

|  |  |  |
| --- | --- | --- |
|  | from pyspark import SparkContext | |
| from pyspark.sql import SQLContext, Row | |
| from pyspark.sql import functions as F | |
| sc = SparkContext(appName = "exercise 2") | |
| sqlContext = SQLContext(sc) | |
| # This path is to the file on hdfs | |
| region\_file = sc.textFile("BDA/input/stations-Ostergotland.csv") | |
| precipitation\_file = sc.textFile("BDA/input/precipitation-readings.csv") | |
| region\_lines = region\_file.map(lambda line: line.split(";")) | |
| pre\_lines = precipitation\_file.map(lambda line: line.split(";")) | |
| regionReadingsRow = region\_lines.map(lambda p: (p[0],)) | |
| preReadingsRow = pre\_lines.map(lambda p: (p[0], p[1], int(p[1].split("-")[0]),int(p[1].split("-")[1]), |  |
| p[2], float(p[3]), p[4] )) | | |
|  | # Specifying the schema programatically and registering the DataFrame as a table | |
| regionReadingsString = ["station"] | |
| preReadingsString = ["station", "date", "year", "month", "time", "pre\_value","quality"] | |
| # Apply the schema to the RDD. | |
| region\_df = sqlContext.createDataFrame(regionReadingsRow,regionReadingsString) | |
| pre\_df = sqlContext.createDataFrame(preReadingsRow,preReadingsString) | |
| # Register the DataFrame as a table. | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | region\_df.registerTempTable("regionstationsTable") | | |
| pre\_df.registerTempTable("preReadingsTable") | | |
| # Can run queries now | | |
| pre\_region\_df = |  | |
| pre\_df.join(region\_df,pre\_df['station']==region\_df['station'],'inner').drop(region\_df['station']) | | | |
|  | avg\_monthly = pre\_region\_df.filter((pre\_region\_df['year']>=1993) & | |  |
| (pre\_region\_df['year']<=2016)).groupBy('year','month','station').agg(F.sum('pre\_value').alias('monthly\_ev | | | |
| erystation')).groupBy('year','month').agg(F.avg('monthly\_everystation').alias('avg\_monthly')).orderBy(['ye | | | |
| ar','month'],ascending=[0,0]) | | | |
| avg\_monthly.rdd.saveAsTextFile("BDA/output") | | | |

First 10 rows of output: