BDA2 Labr Report

Andreas Stasinakis(andst745) & Mim Kemal Tekin(mimte666)

May 10, 2019

Contents

Task 1

In this task we used temperatures-big.csv. First we find minimum and maximum temperatures for each year. We can see results of it as following:

Data import

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

# create Contexts
sc = SparkContext()
sqlContext = SQLContext(sc)

# Load a text file and convert each line to a tuple.
# temperature_file = sc.textFile("../station_data/short_temperature_reads.csv")
# temperature_file = sc.textFile("/user/x_mimte/data/temperatures-big.csv")
temperature_file = sc.textFile("/user/x_mimte/data/temperature-readings.csv")

# transform the data by splitting each line
lines = temperature_file. \
    map(lambda line: line.split(";"))
```

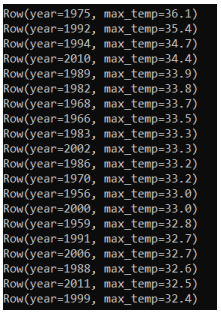
Task 1: Maximum and Minimum Temperature for each Year

```
# import libraries
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row
```

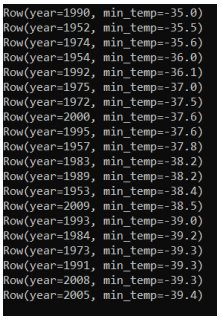
```
from pyspark.sql import functions as F

# create Contexts
sc = SparkContext()
sqlContext = SQLContext(sc)

# Load a text file and convert each line to a tuple.
# temperature_file = sc.textFile("../station_data/short_temperature_reads.csv")
# temperature_file = sc.textFile("/user/x_mimte/data/temperatures-big.csv")
temperature_file = sc.textFile("/user/x_mimte/data/temperature-readings.csv")

# transform the data by splitting each line
lines = temperature_file. \
    map(lambda line: line.split(";"))

# define headers of the dataframe
tempReadingsString= ["station", "date", "year", "month", "time", "value", "quality"]

# map the data for our headers
# ['103100', '1996-07-06', '15:00:00', '14.8', 'G']
tempReadingsRow = lines.map(lambda line: (line[0], line[1], int(line[1][0:4]), \
                            int(line[1][5:7]), line[2], \
                            float(line[3]), line[4]))

# Apply the schema to the RDD.
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)
schemaTempReadings.registerTempTable("tempReadingsTable")


############
#####   1
############
df_year_station = schemaTempReadings. \
    where("year >= 1950 and year <= 2014"). \
    groupBy("year")

df_year_station_max = df_year_station. \
    agg(F.max("value").alias("max_temp")). \
    orderBy("max_temp", ascending=False)

df_year_station_min = df_year_station. \
    agg(F.min("value").alias("min_temp")). \
    orderBy("min_temp", ascending=False)


df_year_station_max.rdd.saveAsTextFile("./bda2_res/t1_max_stations")
df_year_station_min.rdd.saveAsTextFile("./bda2_res/t1_min_stations")


############
#####   1a
############
df_year_station = schemaTempReadings. \
```

```
    where("year >= 1950 and year <= 2014"). \
    groupBy("year", "station")

df_year_station_max = df_year_station. \
    agg(F.max("value").alias("max_temp")). \
    orderBy("max_temp", ascending=False)

df_year_station_min = df_year_station. \
    agg(F.min("value").alias("min_temp")). \
    orderBy("min_temp", ascending=False)

df_year_station_max.rdd.saveAsTextFile("./bda2_res/t1_max_with_stations")
df_year_station_min.rdd.saveAsTextFile("./bda2_res/t1_min_with_stations")
```

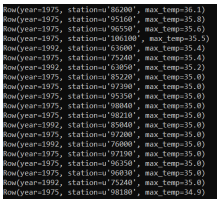
Maximum Temperature



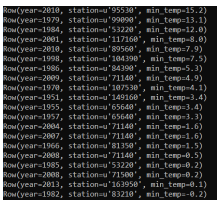
Minimum Temperature



Maximum Temperature with station



Minimum Temperature with station



Task 2

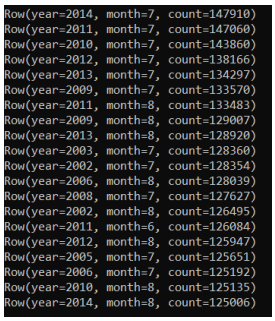
Count the number of readings for each month in the period of 1950-2014 which are higher than 10 degrees. Repeat the exercise, this time taking only distinct readings from each station. That is, if a station reported a reading above 10 degrees in some month, then it appears only once in the count for that month.
In this exercise you will use the temperature-readings.csv file. The output should contain the following information:
Year, month, count

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

# create Contexts
sc = SparkContext()
sqlContext = SQLContext(sc)

# Load a text file and convert each line to a tuple.
# temperature_file = sc.textFile("../station_data/short_temperature_reads.csv")
# temperature_file = sc.textFile("/user/x_mimte/data/temperatures-big.csv")
temperature_file = sc.textFile("/user/x_mimte/data/temperature-readings.csv")

# transform the data by splitting each line
lines = temperature_file. \
    map(lambda line: line.split(";"))

# define headers of the dataframe
tempReadingsString= ["station", "date", "year", "month", "time", "value", "quality"]

# map the data for our headers
# ['103100', '1996-07-06', '15:00:00', '14.8', 'G']
tempReadingsRow = lines.map(lambda line: (line[0], line[1], int(line[1][0:4]), \
                            int(line[1][5:7]), line[2], \
                            float(line[3]), line[4]))

# Apply the schema to the RDD.
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)
schemaTempReadings.registerTempTable("tempReadingsTable")



############
#####   2.i & ii with API
############
counts_monthly_api = schemaTempReadings.\
    where("value>=10 and year>=1950 and year<=2014").\
    groupby("year", "month").\
    agg(F.count("value").alias("count")).\
    orderBy("count", ascending=False)
    # orderBy("count", ascending=False)

counts_monthly_distinct_api = schemaTempReadings.\
    where("value>=10 and year>=1950 and year<=2014").\
    select("year", "month", "station", "value").\
```

```
    distinct().\
    groupby("year", "month").\
    agg(F.count("value").alias("count")).\
    orderBy("count", ascending=False)
    # orderBy("count", ascending=False)

counts_monthly_api.rdd.saveAsTextFile("./bda2_res/t2_api_count_reads_month")
counts_monthly_distinct_api.rdd.saveAsTextFile("./bda2_res/t2_api_count_reads_month_distinct")

#############
#####   2.i & ii with SQL
#############

q1 ="select year, month, count(value) as count \
from tempReadingsTable \
where value>=10 and year between 1950 and 2014 \
group by year, month \
order by count desc"
# order by count desc"

q2 = "SELECT year, month, count(value) as count \
FROM (SELECT distinct year, month, station, value \
FROM tempReadingsTable \
WHERE value >= 10 and year BETWEEN 1950 AND 2014) AS Q \
group by year, month \
order by count desc"
# order by count desc"

counts_monthly = sqlContext.sql(q1)

counts_monthly_distinct = sqlContext.sql(q2)

counts_monthly.rdd.saveAsTextFile("./bda2_res/t2_sql_count_reads_month")
counts_monthly_distinct.rdd.saveAsTextFile("./bda2_res/t2_sql_count_reads_month_distinct")
```

Results with API

Monthly Reading Count Greater than 10



Monthly Distinct Station Reading Count Greater than 10

```
Row(year=2014, month=7, count=39155)
Row(year=2008, month=7, count=34706)
Row(year=2003, month=7, count=34701)
Row(year=2010, month=7, count=34533)
Row(year=2011, month=6, count=34207)
Row(year=1997, month=8, count=34000)
Row(year=2005, month=7, count=33911)
Row(year=2006, month=7, count=33721)
Row(year=1997, month=7, count=33049)
Row(year=2002, month=8, count=32550)
Row(year=2001, month=7, count=32212)
Row(year=2013, month=7, count=32166)
Row(year=2004, month=8, count=32148)
Row(year=2014, month=8, count=32094)
Row(year=1999, month=7, count=31921)
Row(year=2011, month=7, count=31742)
Row(year=2002, month=7, count=31720)
Row(year=1996, month=8, count=31298)
Row(year=2002, month=6, count=31044)
Row(year=1997, month=6, count=30910)
```

Results with SQL

Monthly Reading Count Greater than 10

```
Row(year=2014, month=7, count=147910)
Row(year=2011, month=7, count=147060)
Row(year=2010, month=7, count=143860)
Row(year=2012, month=7, count=138166)
Row(year=2013, month=7, count=134297)
Row(year=2009, month=7, count=133570)
Row(year=2011, month=8, count=133483)
Row(year=2009, month=8, count=129007)
Row(year=2013, month=8, count=128920)
Row(year=2003, month=7, count=128360)
Row(year=2002, month=7, count=128354)
Row(year=2006, month=8, count=128039)
Row(year=2008, month=7, count=127627)
Row(year=2002, month=8, count=126495)
Row(year=2011, month=6, count=126084)
Row(year=2012, month=8, count=125947)
Row(year=2005, month=7, count=125651)
Row(year=2006, month=7, count=125192)
Row(year=2010, month=8, count=125135)
Row(year=2014, month=8, count=125006)
```

Monthly Distinct Station Reading Count Greater than 10

```
Row(year=2014, month=7, count=39155)
Row(year=2008, month=7, count=34706)
Row(year=2003, month=7, count=34701)
Row(year=2010, month=7, count=34533)
Row(year=2011, month=6, count=34207)
Row(year=1997, month=8, count=34000)
Row(year=2005, month=7, count=33911)
Row(year=2006, month=7, count=33721)
Row(year=1997, month=7, count=33049)
Row(year=2002, month=8, count=32550)
Row(year=2001, month=7, count=32212)
Row(year=2013, month=7, count=32166)
Row(year=2004, month=8, count=32148)
Row(year=2014, month=8, count=32094)
Row(year=1999, month=7, count=31921)
Row(year=2011, month=7, count=31742)
Row(year=2002, month=7, count=31720)
Row(year=1996, month=8, count=31298)
Row(year=2002, month=6, count=31044)
Row(year=1997, month=6, count=30910)
```

Task 3

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 number, average monthly temperature

```
# define headers of the dataframe
tempReadingsString= ["station", "date", "year", "month", "time", "value", "quality"]

# map the data for our headers
# ['103100', '1996-07-06', '15:00:00', '14.8', 'G']
tempReadingsRow = lines.map(lambda line: (line[0], line[1], int(line[1][0:4]), \
                            int(line[1][5:7]), line[2], \
```

```
                                float(line[3]), line[4]))

# Apply the schema to the RDD.
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)
schemaTempReadings.registerTempTable("tempReadingsTable")

avg_monthly = schemaTempReadings.\
    where("year>=1960 and year<=2014").\
    groupby("date", "year", "month", "station").\
    agg(((F.max("value") + F.min("value"))/2).alias("avg_daily")).\
    groupby("year", "month", "station").\
    agg(F.avg("avg_daily").alias("avg_monthly")).\
    orderBy("year", "month", ascending=False)
    # orderBy("avg_daily", ascending=False)

avg_monthly.rdd.saveAsTextFile("./bda2_res/t3_avg_month")
```

```
Row(year=2014, month=12, station=u'134090', avg_monthly=-1.8999999999999997
Row(year=2014, month=12, station=u'181900', avg_monthly=-8.606)
Row(year=2014, month=12, station=u'74460', avg_monthly=-0.5193548387096774)
Row(year=2014, month=12, station=u'155900', avg_monthly=-6.853225806451611)
Row(year=2014, month=12, station=u'149340', avg_monthly=-6.461290322580645)
Row(year=2014, month=12, station=u'134590', avg_monthly=-4.685483870967742)
Row(year=2014, month=12, station=u'147560', avg_monthly=-7.093548387096774)
Row(year=2014, month=12, station=u'68560', avg_monthly=3.1983870967741934)
Row(year=2014, month=12, station=u'133250', avg_monthly=-4.951612903225804)
Row(year=2014, month=12, station=u'162870', avg_monthly=-1.3173913043478258
Row(year=2014, month=12, station=u'63160', avg_monthly=1.032258064516129)
Row(year=2014, month=12, station=u'84310', avg_monthly=0.4129032258064519)
Row(year=2014, month=12, station=u'85490', avg_monthly=-0.3951612903225806)
Row(year=2014, month=12, station=u'93220', avg_monthly=-1.6693548387096775)
Row(year=2014, month=12, station=u'83420', avg_monthly=1.382258064516129)
Row(year=2014, month=12, station=u'158750', avg_monthly=-7.120967741935485)
Row(year=2014, month=12, station=u'157860', avg_monthly=-8.911290322580646)
Row(year=2014, month=12, station=u'97120', avg_monthly=-0.7080645161290322)
Row(year=2014, month=12, station=u'104580', avg_monthly=-4.216129032258065)
Row(year=2014, month=12, station=u'54300', avg_monthly=2.5919354838709676)
```

Task 4

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 200 mm. In this exercise you will use the temperature-readings.csv and precipitation-readings.csv files.
The output should contain the following information:

Station number, maximum measured temperature, maximum daily precipitation

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

# create Contexts
sc = SparkContext()
sqlContext = SQLContext(sc)

# Load a text file and convert each line to a tuple.
# temperature_file = sc.textFile("../station_data/short_temperature_reads.csv")
# temperature_file = sc.textFile("/user/x_mimte/data/temperatures-big.csv")
temperature_file = sc.textFile("/user/x_mimte/data/temperature-readings.csv")

# transform the data by splitting each line
lines = temperature_file. \
    map(lambda line: line.split(";"))
```

```
# define headers of the dataframe
tempReadingsString = ["station", "date", "year", "month", "time", "value", "quality"]

# map the data for our headers
# ['103100', '1996-07-06', '15:00:00', '14.8', 'G']
tempReadingsRow = lines.map(lambda line: (line[0], line[1], int(line[1][0:4]), \
                            int(line[1][5:7]), line[2], \
                            float(line[3]), line[4]))


# Apply the schema to the RDD.
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)
schemaTempReadings.registerTempTable("tempReadingsTable")

# import the data
# precipitation_file = sc.textFile("../station_data/short_precipitation-readings.csv")
precipitation_file = sc.textFile("/user/x_mimte/data/precipitation-readings.csv")

# transform the data by splitting each line
lines_precipitation = precipitation_file. \
    map(lambda line: line.split(";"))

# define headers of the dataframe
precipitationString= ["station", "date", "year", "month", "time", "value", "quality"]

precipitationRow = lines_precipitation.map(lambda line: (line[0], line[1], int(line[1][0:4]), \
                            int(line[1][5:7]), line[2], \
                            float(line[3]), line[4]))

schemaPrecipitation = sqlContext.createDataFrame(precipitationRow, precipitationString)
schemaPrecipitation.registerTempTable("precipitationtable")



max_temp = schemaTempReadings.\
    groupby("station").\
    agg(F.max("value").alias("max_temp")).\
    where("max_temp>=25 and max_temp<=30")



max_daily_prec = schemaPrecipitation.\
    groupby("date", "station").\
    agg(F.sum("value").alias("daily_prec")).\
    groupby("station").\
    agg(F.max("daily_prec").alias("max_prec")).\
    where("max_prec>100 and max_prec<200")

max_values = max_temp.\
    join(max_daily_prec, "station")

max_values.rdd.saveAsTextFile("./bda2_res/t4_station_temp_prec")
```

The output is an empty file.

Task 5

Calculate the average monthly precipitation for the Ostergotland 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).
In this exercise you will use the precipitation-readings.csv and stations-Ostergotland.csv files.
The output should contain the following information:
Year, month, average monthly precipitation

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

# create Contexts
sc = SparkContext()
sqlContext = SQLContext(sc)

# Load a text file and convert each line to a tuple.
# temperature_file = sc.textFile("../station_data/short_temperature_reads.csv")
# temperature_file = sc.textFile("/user/x_mimte/data/temperatures-big.csv")
temperature_file = sc.textFile("/user/x_mimte/data/temperature-readings.csv")

# transform the data by splitting each line
lines = temperature_file. \
    map(lambda line: line.split(";"))

# define headers of the dataframe
tempReadingsString = ["station", "date", "year", "month", "time", "value", "quality"]

# map the data for our headers
# ['103100', '1996-07-06', '15:00:00', '14.8', 'G']
tempReadingsRow = lines.map(lambda line: (line[0], line[1], int(line[1][0:4]), \
                            int(line[1][5:7]), line[2], \
                            float(line[3]), line[4]))


# Apply the schema to the RDD.
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)
schemaTempReadings.registerTempTable("tempReadingsTable")

# import the data
# precipitation_file = sc.textFile("../station_data/short_precipitation-readings.csv")
precipitation_file = sc.textFile("/user/x_mimte/data/precipitation-readings.csv")

# transform the data by splitting each line
lines_precipitation = precipitation_file. \
    map(lambda line: line.split(";"))

# define headers of the dataframe
precipitationString= ["station", "date", "year", "month", "time", "value", "quality"]
```

```
precipitationRow = lines_precipitation.map(lambda line: (line[0], line[1], int(line[1][0:4]), \
                           int(line[1][5:7]), line[2], \
                           float(line[3]), line[4]))

schemaPrecipitation = sqlContext.createDataFrame(precipitationRow, precipitationString)
schemaPrecipitation.registerTempTable("precipitationtable")


# import stations in Ostergotland
# ost_station_file = sc.textFile("../station_data/stations-Ostergotland.csv")
ost_station_file = sc.textFile("/user/x_mimte/data/stations-Ostergotland.csv")

# transform the data by splitting each line
lines_ost = ost_station_file. \
    map(lambda line: line.split(";"))

# map the data for our headers
# We could add all other features too. But we do not have any question
# which asks for all other informations.
ostStationsString = ["station", "station_name"]

ostStationsRow = lines_ost.map(lambda line: (line[0], line[1]))


schemaOstStations = sqlContext.createDataFrame(ostStationsRow, ostStationsString)
schemaOstStations.registerTempTable("oststations")



############
#####   5
############

ostStations = schemaOstStations.select("station")

#another way
# avg_monthly_long = schemaOstStations.\
#     join(schemaPrecipitation, "station").\
#     where("year>1993 and year<2016").\
#     groupby("year", "month", "station").\
#     agg(F.avg("value").alias("avg_prec")).\
#     groupBy("year", "month").\
#     agg(F.avg("avg_prec").alias("avg_prec")).\
#     orderBy("year", "month", ascending=False)
#

avg_monthly = schemaOstStations.\
    join(schemaPrecipitation, "station").\
    where("year>1993 and year<2016").\
    groupby("year", "month").\
    agg(F.avg("value").alias("avg_prec")).\
    orderBy("year", "month", ascending=False)
```

```
# avg_monthly_long.rdd.saveAsTextFile("./bda2_res/t5_avg_ost_station_long")
avg_monthly.rdd.saveAsTextFile("./bda2_res/t5_avg_ost_station")
```

```
Row(year=2015, month=12, avg_prec=0.04152907394113425)
Row(year=2015, month=11, avg_prec=0.09301182893539582)
Row(year=2015, month=10, avg_prec=0.0032018397311162215)
Row(year=2015, month=9, avg_prec=0.1466787330316742)
Row(year=2015, month=8, avg_prec=0.03825301204819278)
Row(year=2015, month=7, avg_prec=0.16872675757039132)
Row(year=2015, month=6, avg_prec=0.11523530488921445)
Row(year=2015, month=5, avg_prec=0.13079621185548929)
Row(year=2015, month=4, avg_prec=0.022284780239738466)
Row(year=2015, month=3, avg_prec=0.05980701754385966)
Row(year=2015, month=2, avg_prec=0.038925911407291264)
Row(year=2015, month=1, avg_prec=0.08375841303577759)
Row(year=2014, month=12, avg_prec=0.050399715757683435)
Row(year=2014, month=11, avg_prec=0.07664473684210535)
Row(year=2014, month=10, avg_prec=0.1020152024041011)
Row(year=2014, month=9, avg_prec=0.07054969057153258)
Row(year=2014, month=8, avg_prec=0.12998747539810337)
Row(year=2014, month=7, avg_prec=0.03264690218356116)
Row(year=2014, month=6, avg_prec=0.10947004188672374)
Row(year=2014, month=5, avg_prec=0.08136068735753114)
```

Task 6

Compare the average monthly temperature (find the difference) in the period 1950-2014 for all stations in Ostergotland with long-term monthly averages in the period of 1950-1980. Make a plot of your results. The output should contain the following information:

Year, month, difference

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

# create Contexts
sc = SparkContext()
sqlContext = SQLContext(sc)

# Load a text file and convert each line to a tuple.
# temperature_file = sc.textFile("../station_data/short_temperature_reads.csv")
# temperature_file = sc.textFile("/user/x_mimte/data/temperatures-big.csv")
temperature_file = sc.textFile("/user/x_mimte/data/temperature-readings.csv")

# transform the data by splitting each line
lines = temperature_file. \
    map(lambda line: line.split(";"))

# define headers of the dataframe
tempReadingsString= ["station", "date", "year", "month", "time", "value", "quality"]

# map the data for our headers
# ['103100', '1996-07-06', '15:00:00', '14.8', 'G']
tempReadingsRow = lines.map(lambda line: (line[0], line[1], int(line[1][0:4]), \
                            int(line[1][5:7]), line[2], \
                            float(line[3]), line[4]))

# Apply the schema to the RDD.
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)
schemaTempReadings.registerTempTable("tempReadingsTable")

# import stations in Ostergotland
# ost_station_file = sc.textFile("../station_data/stations-Ostergotland.csv")
```

```
ost_station_file = sc.textFile("/user/x_mimte/data/stations-Ostergotland.csv")

# transform the data by splitting each line
lines_ost = ost_station_file. \
    map(lambda line: line.split(";"))

# map the data for our headers
# We could add all other features too. But we do not have any question
# which asks for all other informations.
ostStationsString = ["station", "station_name"]

ostStationsRow = lines_ost.map(lambda line: (line[0], line[1]))


schemaOstStations = sqlContext.createDataFrame(ostStationsRow, ostStationsString)
schemaOstStations.registerTempTable("oststations")


############
#####   6
############
avg_monthly50_14 = schemaOstStations.\
    join(schemaTempReadings, "station").\
    where("year>=1950 and year<=2014").\
    groupby("date", "year", "month", "station").\
    agg(((F.max("value") + F.min("value"))/2).alias("avg_daily")).\
    groupby("year", "month").\
    agg(F.avg("avg_daily").alias("avg_monthly")).\
    orderBy("year", "month", ascending=False)
    # orderBy("avg_daily", ascending=False)

long_term_avg = avg_monthly50_14.\
    where("year<=1980").\
    groupby("month").\
    agg(F.avg("avg_monthly").alias("avg")).\
    orderBy("month", ascending=False)


avg_monthly50_14.rdd.saveAsTextFile("./bda2_res/t6_avg_monthly")
long_term_avg.rdd.saveAsTextFile("./bda2_res/t6_long_term_avg")
```

Average Monthly

```
Row(year=2014, month=12, avg_monthly=-0.16581469648562294)
Row(year=2014, month=11, avg_monthly=4.380500000000005)
Row(year=2014, month=10, avg_monthly=8.726832844574781)
Row(year=2014, month=9, avg_monthly=11.869545454545454)
Row(year=2014, month=8, avg_monthly=15.443255131964808)
Row(year=2014, month=7, avg_monthly=19.029032258064515)
Row(year=2014, month=6, avg_monthly=13.773030303030302)
Row(year=2014, month=5, avg_monthly=10.580806451612904)
Row(year=2014, month=4, avg_monthly=6.544666666666667)
Row(year=2014, month=3, avg_monthly=3.8316129032258064)
Row(year=2014, month=2, avg_monthly=1.5955357142857143)
Row(year=2014, month=1, avg_monthly=-2.2908064516129034)
Row(year=2013, month=12, avg_monthly=2.8829268292682926)
Row(year=2013, month=11, avg_monthly=3.251212121212121)
Row(year=2013, month=10, avg_monthly=7.957184750733137)
Row(year=2013, month=9, avg_monthly=10.83276397515528)
Row(year=2013, month=8, avg_monthly=15.771260997067445)
Row(year=2013, month=7, avg_monthly=16.943401759530794)
Row(year=2013, month=6, avg_monthly=15.036212121212121)
Row(year=2013, month=5, avg_monthly=11.887536656891495)
```

Long Term Average Monthly

```
Row(month=12, avg=-0.9596664798954081)
Row(month=11, avg=2.316960327307101)
Row(month=10, avg=7.204875353956812)
Row(month=9, avg=11.808487268108237)
Row(month=8, avg=16.085902203935504)
Row(month=7, avg=16.92311041935054)
Row(month=6, avg=15.580398922761828)
Row(month=5, avg=10.313615801472201)
Row(month=4, avg=4.47847350767512)
Row(month=3, avg=-0.6551139529911643)
Row(month=2, avg=-3.8247756002803284)
Row(month=1, avg=-3.2233944723330774)
```