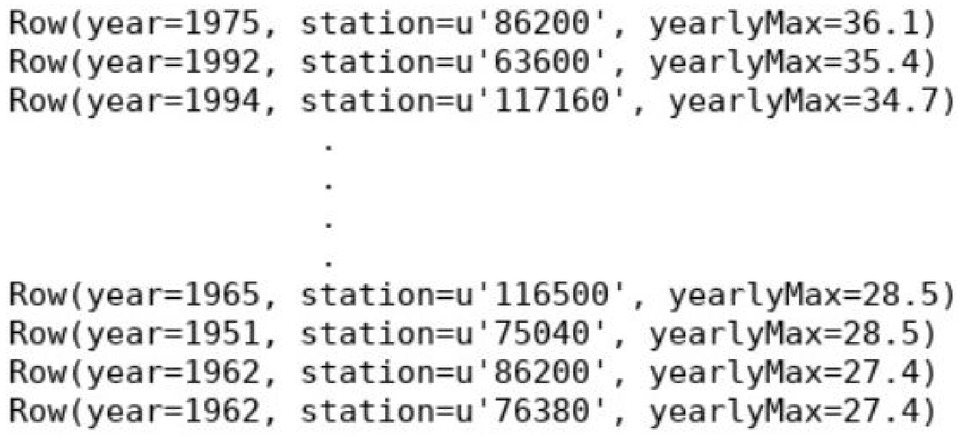
wilan057
chrvo878
2020-05-15

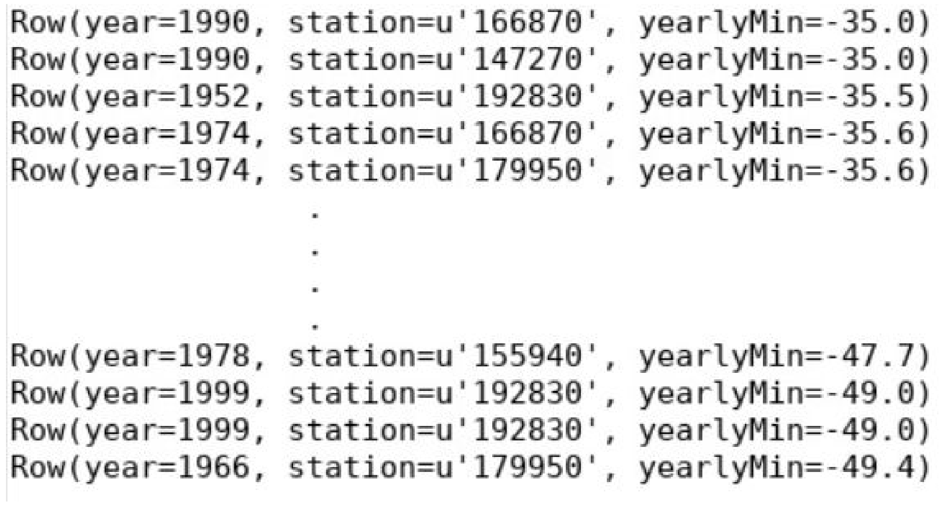
Lab report 2 - Big Data Analytics TDDE31

1) The output should contain the following information:
year, station with the max, maxValue ORDER BY maxValue DESC



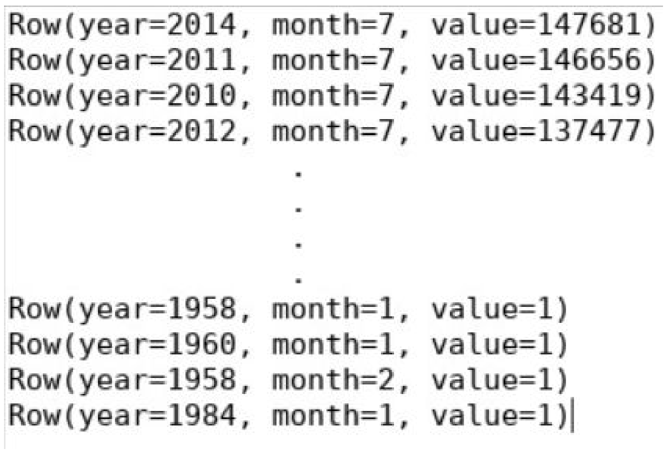
```
Row(year=1975, station=u'86200', yearlyMax=36.1)
Row(year=1992, station=u'63600', yearlyMax=35.4)
Row(year=1994, station=u'117160', yearlyMax=34.7)
                    .
                    .
                    .
                    .
Row(year=1965, station=u'116500', yearlyMax=28.5)
Row(year=1951, station=u'75040', yearlyMax=28.5)
Row(year=1962, station=u'86200', yearlyMax=27.4)
Row(year=1962, station=u'76380', yearlyMax=27.4)
```

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



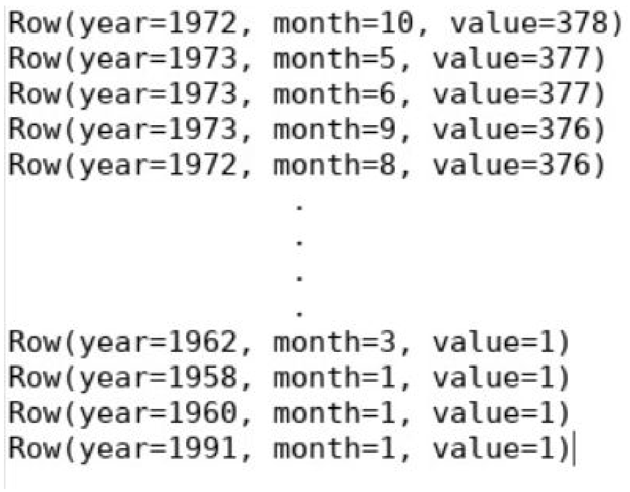
```
Row(year=1990, station=u'166870', yearlyMin=-35.0)
Row(year=1990, station=u'147270', yearlyMin=-35.0)
Row(year=1952, station=u'192830', yearlyMin=-35.5)
Row(year=1974, station=u'166870', yearlyMin=-35.6)
Row(year=1974, station=u'179950', yearlyMin=-35.6)
                    .
                    .
                    .
                    .
Row(year=1978, station=u'155940', yearlyMin=-47.7)
Row(year=1999, station=u'192830', yearlyMin=-49.0)
Row(year=1999, station=u'192830', yearlyMin=-49.0)
Row(year=1966, station=u'179950', yearlyMin=-49.4)
```

wilan057
chrvo878
2020-05-15
2) The output should contain the following information:
year, month, value ORDER BY value DESC



```
Row(year=2014, month=7, value=147681)
Row(year=2011, month=7, value=146656)
Row(year=2010, month=7, value=143419)
Row(year=2012, month=7, value=137477)
                 .
                 .
                 .
                 .
Row(year=1958, month=1, value=1)
Row(year=1960, month=1, value=1)
Row(year=1958, month=2, value=1)
Row(year=1984, month=1, value=1)
```

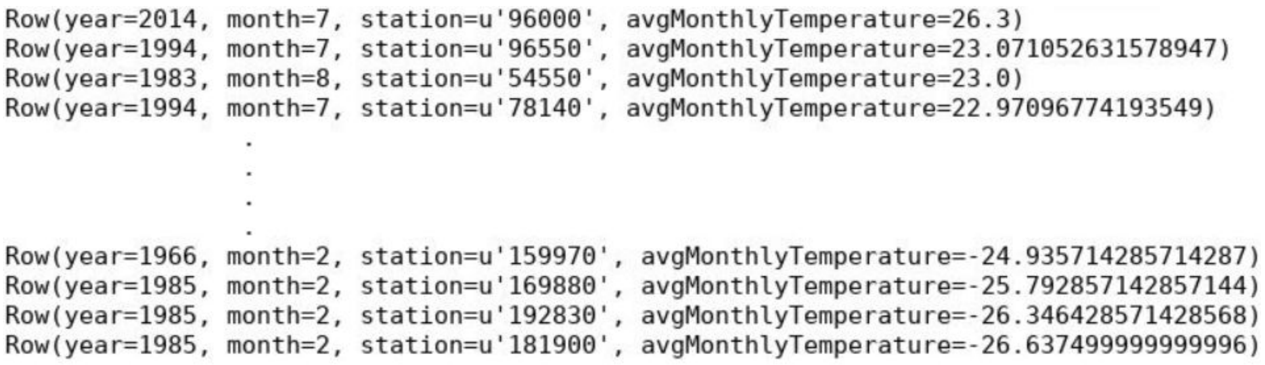
Count of instances above 10



```
Row(year=1972, month=10, value=378)
Row(year=1973, month=5, value=377)
Row(year=1973, month=6, value=377)
Row(year=1973, month=9, value=376)
Row(year=1972, month=8, value=376)
                 .
                 .
                 .
                 .
Row(year=1962, month=3, value=1)
Row(year=1958, month=1, value=1)
Row(year=1960, month=1, value=1)
Row(year=1991, month=1, value=1)
```

Count of Distinct instances above 10

wilan057
chrvo878
2020-05-15
3) The output should contain the following information:
year, month, station, avgMonthlyTemperature ORDER BY avgMonthlyTemperature DESC

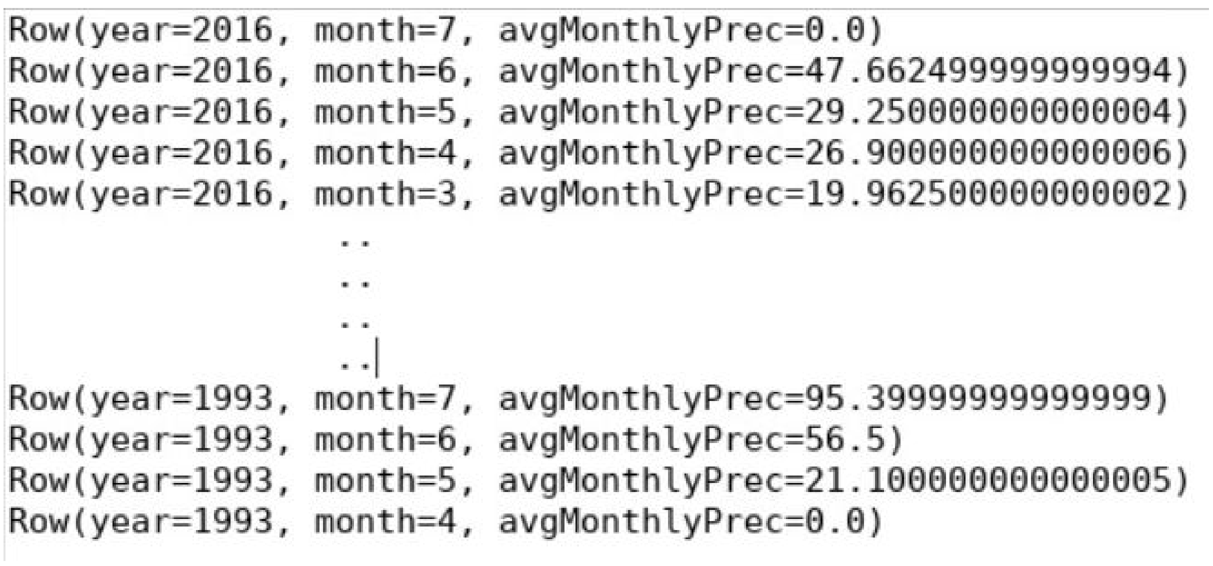


```
Row(year=2014, month=7, station=u'96000', avgMonthlyTemperature=26.3)
Row(year=1994, month=7, station=u'96550', avgMonthlyTemperature=23.071052631578947)
Row(year=1983, month=8, station=u'54550', avgMonthlyTemperature=23.0)
Row(year=1994, month=7, station=u'78140', avgMonthlyTemperature=22.97096774193549)
                 .
                 .
                 .
                 .
Row(year=1966, month=2, station=u'159970', avgMonthlyTemperature=-24.935714285714287)
Row(year=1985, month=2, station=u'169880', avgMonthlyTemperature=-25.792857142857144)
Row(year=1985, month=2, station=u'192830', avgMonthlyTemperature=-26.346428571428568)
Row(year=1985, month=2, station=u'181900', avgMonthlyTemperature=-26.637499999999996)
```

4) The output should contain the following information:
station, maxTemp, maxDailyPrecipitation ORDER BY station DESC

EMPTY OUTPUT

5) The output should contain the following information:
year, month, avgMonthlyPrecipitation ORDER BY year DESC, month DESC



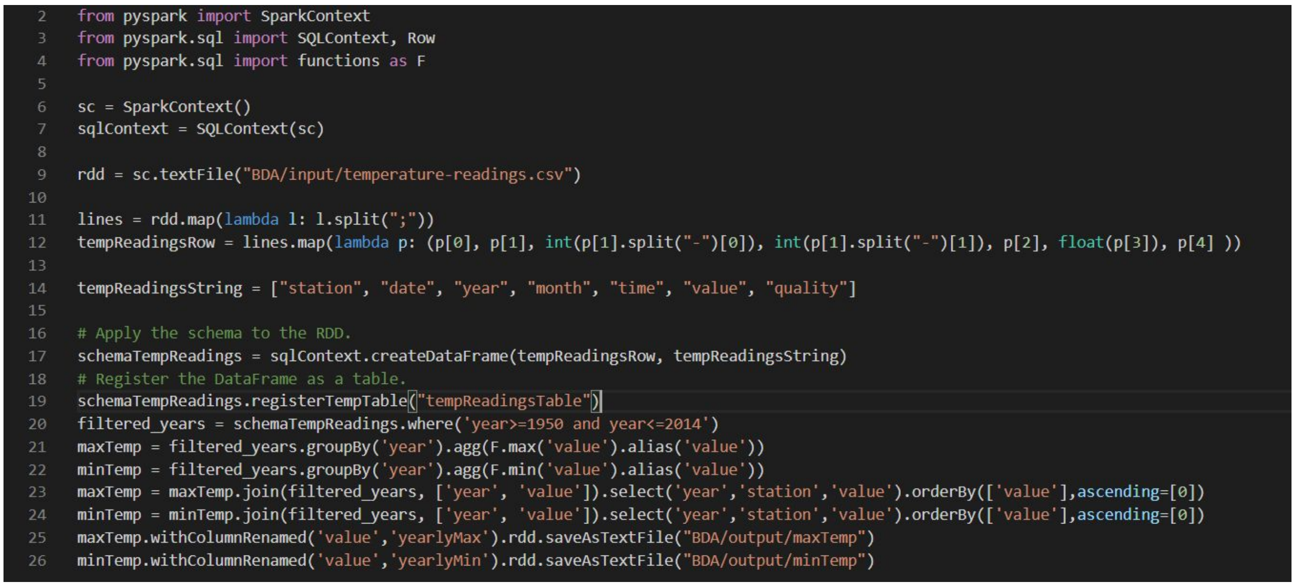
```
Row(year=2016, month=7, avgMonthlyPrec=0.0)
Row(year=2016, month=6, avgMonthlyPrec=47.662499999999994)
Row(year=2016, month=5, avgMonthlyPrec=29.250000000000004)
Row(year=2016, month=4, avgMonthlyPrec=26.900000000000006)
Row(year=2016, month=3, avgMonthlyPrec=19.962500000000002)
                   ..
                   ..
                   ..
                   ..|
Row(year=1993, month=7, avgMonthlyPrec=95.39999999999999)
Row(year=1993, month=6, avgMonthlyPrec=56.5)
Row(year=1993, month=5, avgMonthlyPrec=21.100000000000005)
Row(year=1993, month=4, avgMonthlyPrec=0.0)
```

wilan057
chrvo878
2020-05-15

Appendix

Exercise 1

Picture:



```python
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")

lines = rdd.map(lambda l: l.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] ))

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

# Apply the schema to the RDD.
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)
# Register the DataFrame as a table.
schemaTempReadings.registerTempTable("tempReadingsTable")
filtered_years = schemaTempReadings.where('year>=1950 and year<=2014')
maxTemp = filtered_years.groupBy('year').agg(F.max('value').alias('value'))
minTemp = filtered_years.groupBy('year').agg(F.min('value').alias('value'))
maxTemp = maxTemp.join(filtered_years, ['year', 'value']).select('year','station','value').orderBy(['value'],ascending=[0])
minTemp = minTemp.join(filtered_years, ['year', 'value']).select('year','station','value').orderBy(['value'],ascending=[0])
maxTemp.withColumnRenamed('value','yearlyMax').rdd.saveAsTextFile("BDA/output/maxTemp")
minTemp.withColumnRenamed('value','yearlyMin').rdd.saveAsTextFile("BDA/output/minTemp")
```

Text:

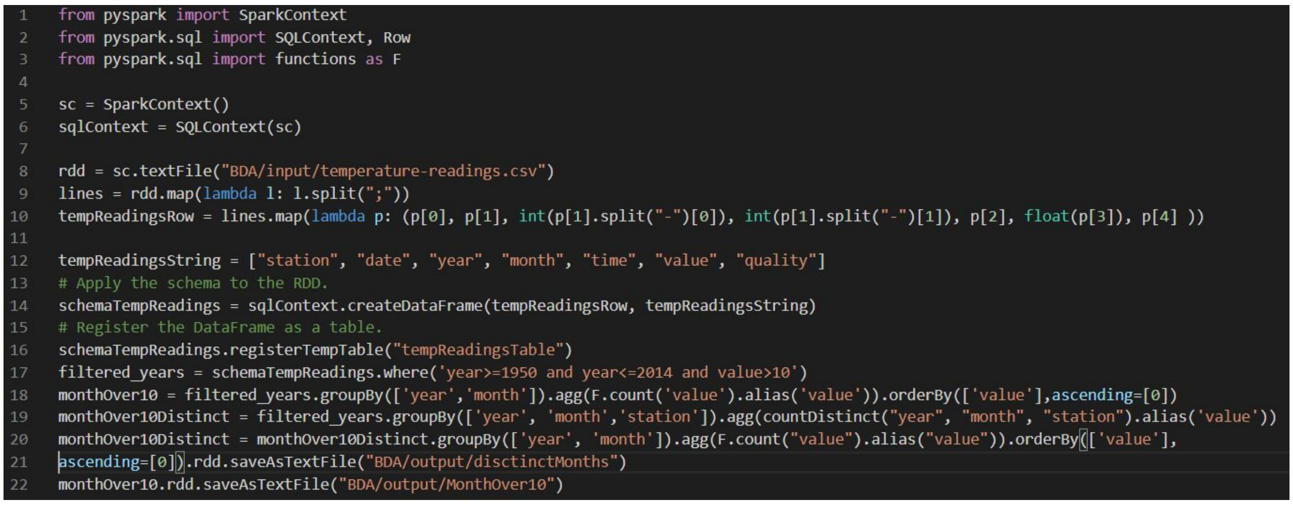
```
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row
from pyspark.sql import functions as F

sc = SparkContext()
sqlContext = SQLContext(sc)

rdd = sc.textFile("BDA/input/temperature-readings.csv")

lines = rdd.map(lambda l: l.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] ))

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

# Apply the schema to the RDD.
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow,
tempReadingsString)
# Register the DataFrame as a table.
schemaTempReadings.registerTempTable("tempReadingsTable")
filtered_years = schemaTempReadings.where('year>=1950 and year<=2014')
maxTemp = filtered_years.groupBy('year').agg(F.max('value').alias('value'))
```

wilan057
chrvo878
2020-05-15

```
minTemp = filtered_years.groupBy('year').agg(F.min('value').alias('value'))
maxTemp = maxTemp.join(filtered_years, ['year',
'value']).select('year','station','value').orderBy(['value'],ascending=[0])
minTemp = minTemp.join(filtered_years, ['year',
'value']).select('year','station','value').orderBy(['value'],ascending=[0])
maxTemp.withColumnRenamed('value','yearlyMax').rdd.saveAsTextFile("BDA/output/maxTe
mp")
minTemp.withColumnRenamed('value','yearlyMin').rdd.saveAsTextFile("BDA/output/minTem
p")
```

Exercise 2:

Picture:



```
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")
lines = rdd.map(lambda l: l.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] ))

tempReadingsString = ["station", "date", "year", "month", "time", "value", "quality"]
# Apply the schema to the RDD.
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)
# Register the DataFrame as a table.
schemaTempReadings.registerTempTable("tempReadingsTable")
filtered_years = schemaTempReadings.where('year>=1950 and year<=2014 and value>10')
monthOver10 = filtered_years.groupBy(['year','month']).agg(F.count('value').alias('value')).orderBy(['value'],ascending=[0])
monthOver10Distinct = filtered_years.groupBy(['year', 'month','station']).agg(countDistinct("year", "month", "station").alias('value'))
monthOver10Distinct = monthOver10Distinct.groupBy(['year', 'month']).agg(F.count("value").alias("value")).orderBy(['value'],
ascending=[0]).rdd.saveAsTextFile("BDA/output/disctinctMonths")
monthOver10.rdd.saveAsTextFile("BDA/output/MonthOver10")
```

Text:

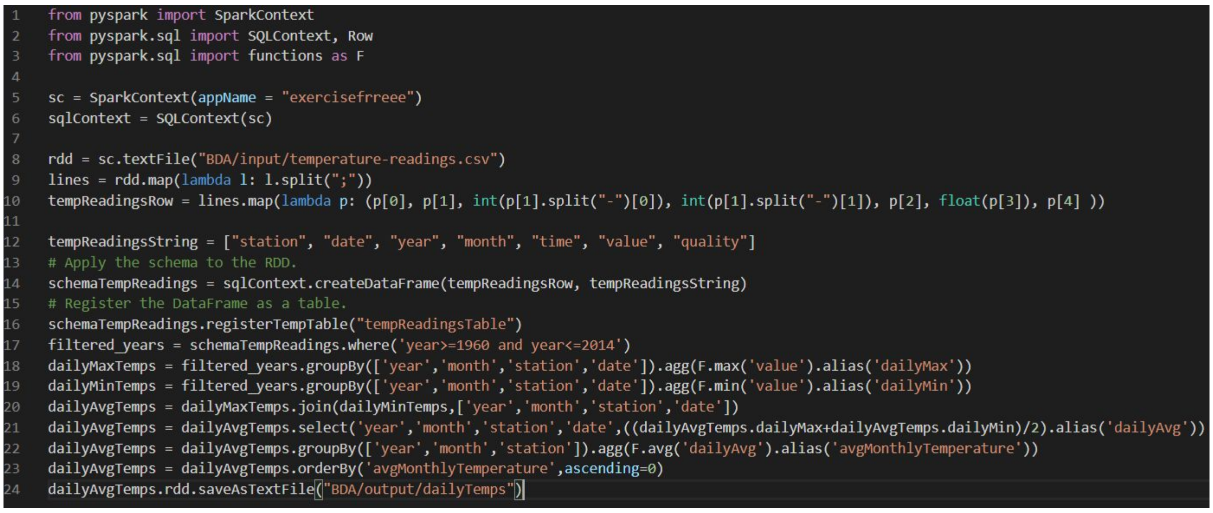
```
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row
from pyspark.sql import functions as F

sc = SparkContext()
sqlContext = SQLContext(sc)

rdd = sc.textFile("BDA/input/temperature-readings.csv")
lines = rdd.map(lambda l: l.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] ))

tempReadingsString = ["station", "date", "year", "month", "time", "value", "quality"]
# Apply the schema to the RDD.
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow,
tempReadingsString)
```

wilan057
chrvo878
2020-05-15

```
# Register the DataFrame as a table.
schemaTempReadings.registerTempTable("tempReadingsTable")
filtered_years = schemaTempReadings.where('year>=1950 and year<=2014 and value>10')
monthOver10 = filtered_years.groupBy(['year','month']).agg(F.count('value').alias('value'))
monthOver10Distinct = filtered_years.groupBy(['year',
'month','station']).agg(F.countDistinct('year', 'month', 'station').alias('value'))
monthOver10Distinct = monthOver10Distinct.groupBy(['year',
'month']).agg(F.count('value').alias('value'))
monthOver10Distinct.orderBy('value',ascending=0).rdd.saveAsTextFile("BDA/output/disctinct
Months")
monthOver10.orderBy(['value'],ascending=[0]).rdd.saveAsTextFile("BDA/output/MonthOver1
0")
```

Exercise 3:

Picture:



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

sc = SparkContext(appName = "exercisefrreee")
sqlContext = SQLContext(sc)

rdd = sc.textFile("BDA/input/temperature-readings.csv")
lines = rdd.map(lambda l: l.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] ))

tempReadingsString = ["station", "date", "year", "month", "time", "value", "quality"]
# Apply the schema to the RDD.
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)
# Register the DataFrame as a table.
schemaTempReadings.registerTempTable("tempReadingsTable")
filtered_years = schemaTempReadings.where('year>=1960 and year<=2014')
dailyMaxTemps = filtered_years.groupBy(['year','month','station','date']).agg(F.max('value').alias('dailyMax'))
dailyMinTemps = filtered_years.groupBy(['year','month','station','date']).agg(F.min('value').alias('dailyMin'))
dailyAvgTemps = dailyMaxTemps.join(dailyMinTemps,['year','month','station','date'])
dailyAvgTemps = dailyAvgTemps.select('year','month','station','date',((dailyAvgTemps.dailyMax+dailyAvgTemps.dailyMin)/2).alias('dailyAvg'))
dailyAvgTemps = dailyAvgTemps.groupBy(['year','month','station']).agg(F.avg('dailyAvg').alias('avgMonthlyTemperature'))
dailyAvgTemps = dailyAvgTemps.orderBy('avgMonthlyTemperature',ascending=0)
dailyAvgTemps.rdd.saveAsTextFile("BDA/output/dailyTemps")
```

Text:

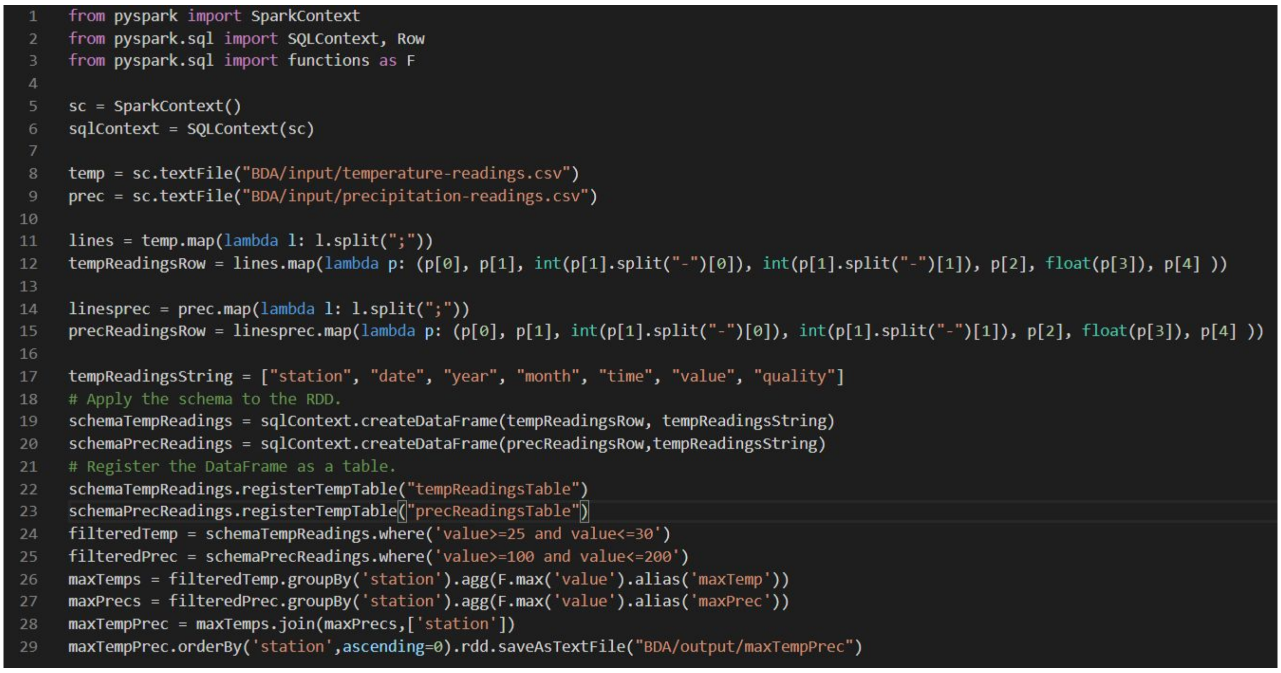
```
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row
from pyspark.sql import functions as F

sc = SparkContext(appName = "exercisefrreee")
sqlContext = SQLContext(sc)

rdd = sc.textFile("BDA/input/temperature-readings.csv")
lines = rdd.map(lambda l: l.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] ))

tempReadingsString = ["station", "date", "year", "month", "time", "value", "quality"]
```

wilan057
chrvo878
2020-05-15

```
# Apply the schema to the RDD.
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow,
tempReadingsString)
# Register the DataFrame as a table.
schemaTempReadings.registerTempTable("tempReadingsTable")
filtered_years = schemaTempReadings.where('year>=1960 and year<=2014')
dailyMaxTemps =
filtered_years.groupBy(['year','month','station','date']).agg(F.max('value').alias('dailyMax'))
dailyMinTemps =
filtered_years.groupBy(['year','month','station','date']).agg(F.min('value').alias('dailyMin'))
dailyAvgTemps = dailyMaxTemps.join(dailyMinTemps,['year','month','station','date'])
dailyAvgTemps =
dailyAvgTemps.select('year','month','station','date',((dailyAvgTemps.dailyMax+dailyAvgTemp
s.dailyMin)/2).alias('dailyAvg'))
dailyAvgTemps =
dailyAvgTemps.groupBy(['year','month','station']).agg(F.avg('dailyAvg').alias('avgMonthlyTem
perature'))
dailyAvgTemps = dailyAvgTemps.orderBy('avgMonthlyTemperature',ascending=0)
dailyAvgTemps.rdd.saveAsTextFile("BDA/output/dailyTemps")
```

Exercise 4:

Picture:



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

sc = SparkContext()
sqlContext = SQLContext(sc)

temp = sc.textFile("BDA/input/temperature-readings.csv")
prec = sc.textFile("BDA/input/precipitation-readings.csv")

lines = temp.map(lambda l: l.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] ))

linesprec = prec.map(lambda l: l.split(";"))
precReadingsRow = linesprec.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"]
# Apply the schema to the RDD.
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow, tempReadingsString)
schemaPrecReadings = sqlContext.createDataFrame(precReadingsRow,tempReadingsString)
# Register the DataFrame as a table.
schemaTempReadings.registerTempTable("tempReadingsTable")
schemaPrecReadings.registerTempTable("precReadingsTable")
filteredTemp = schemaTempReadings.where('value>=25 and value<=30')
filteredPrec = schemaPrecReadings.where('value>=100 and value<=200')
maxTemps = filteredTemp.groupBy('station').agg(F.max('value').alias('maxTemp'))
maxPrecs = filteredPrec.groupBy('station').agg(F.max('value').alias('maxPrec'))
maxTempPrec = maxTemps.join(maxPrecs,['station'])
maxTempPrec.orderBy('station',ascending=0).rdd.saveAsTextFile("BDA/output/maxTempPrec")
```

Text:

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

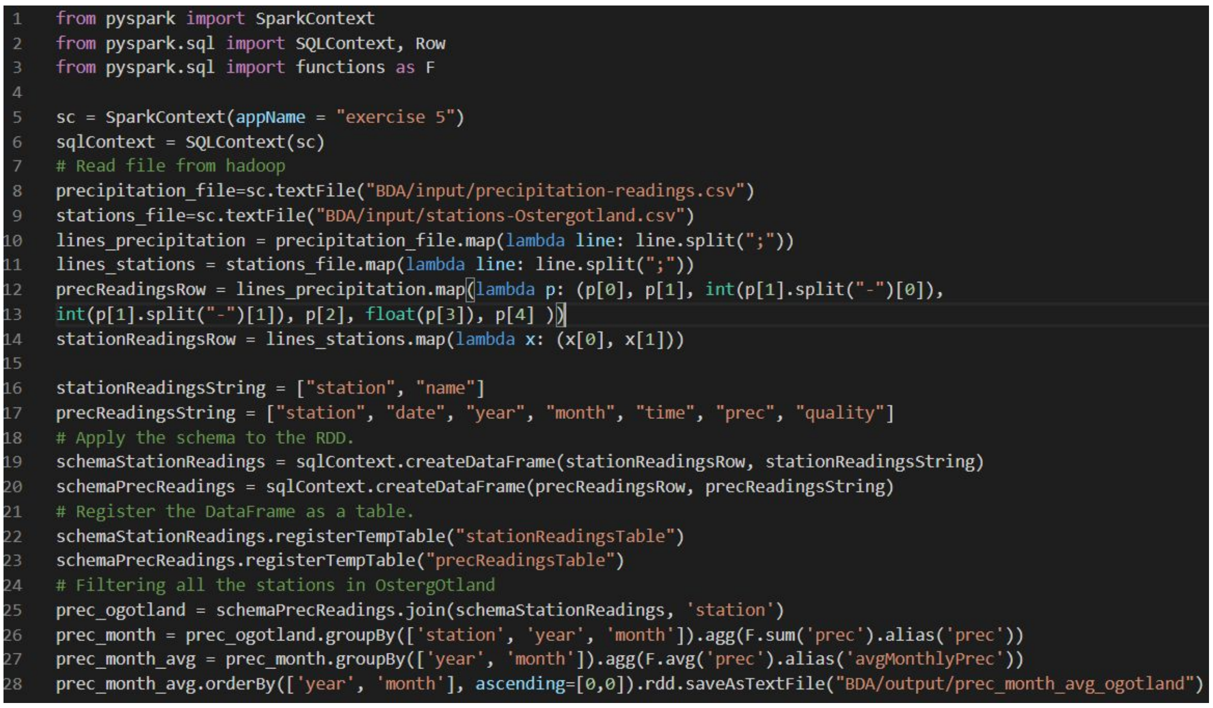
wilan057
chrvo878
2020-05-15

```
from pyspark.sql import functions as F

sc = SparkContext()
sqlContext = SQLContext(sc)

temp = sc.textFile("BDA/input/temperature-readings.csv")
prec = sc.textFile("BDA/input/precipitation-readings.csv")

lines = temp.map(lambda l: l.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] ))

linesprec = prec.map(lambda l: l.split(";"))
precReadingsRow = linesprec.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"]
# Apply the schema to the RDD.
schemaTempReadings = sqlContext.createDataFrame(tempReadingsRow,
tempReadingsString)
schemaPrecReadings =
sqlContext.createDataFrame(precReadingsRow,tempReadingsString)
# Register the DataFrame as a table.
schemaTempReadings.registerTempTable("tempReadingsTable")
schemaPrecReadings.registerTempTable("precReadingsTable")
filteredTemp = schemaTempReadings.where('value>=25 and value<=30')
filteredPrec = schemaPrecReadings.where('value>=100 and value<=200')
maxTemps = filteredTemp.groupBy('station').agg(F.max('value').alias('maxTemp'))
maxPrecs = filteredPrec.groupBy('station').agg(F.max('value').alias('maxPrec'))
maxTempPrec = maxTemps.join(maxPrecs,['station'])
maxTempPrec.orderBy('station',ascending=0).rdd.saveAsTextFile("BDA/output/maxTempPr
ec")
```

wilan057
chrvo878
2020-05-15

Exercise 5:

Picture:



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

sc = SparkContext(appName = "exercise 5")
sqlContext = SQLContext(sc)
# Read file from hadoop
precipitation_file=sc.textFile("BDA/input/precipitation-readings.csv")
stations_file=sc.textFile("BDA/input/stations-Ostergotland.csv")
lines_precipitation = precipitation_file.map(lambda line: line.split(";"))
lines_stations = stations_file.map(lambda line: line.split(";"))
precReadingsRow = lines_precipitation.map(lambda p: (p[0], p[1], int(p[1].split("-")[0]),
int(p[1].split("-")[1]), p[2], float(p[3]), p[4] ))
stationReadingsRow = lines_stations.map(lambda x: (x[0], x[1]))

stationReadingsString = ["station", "name"]
precReadingsString = ["station", "date", "year", "month", "time", "prec", "quality"]
# Apply the schema to the RDD.
schemaStationReadings = sqlContext.createDataFrame(stationReadingsRow, stationReadingsString)
schemaPrecReadings = sqlContext.createDataFrame(precReadingsRow, precReadingsString)
# Register the DataFrame as a table.
schemaStationReadings.registerTempTable("stationReadingsTable")
schemaPrecReadings.registerTempTable("precReadingsTable")
# Filtering all the stations in Ostergotland
prec_ogotland = schemaPrecReadings.join(schemaStationReadings, 'station')
prec_month = prec_ogotland.groupBy(['station', 'year', 'month']).agg(F.sum('prec').alias('prec'))
prec_month_avg = prec_month.groupBy(['year', 'month']).agg(F.avg('prec').alias('avgMonthlyPrec'))
prec_month_avg.orderBy(['year', 'month'], ascending=[0,0]).rdd.saveAsTextFile("BDA/output/prec_month_avg_ogotland")
```

Text:

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

sc = SparkContext(appName = "exercise 5")
sqlContext = SQLContext(sc)
# Read file from hadoop
precipitation_file=sc.textFile("BDA/input/precipitation-readings.csv")
stations_file=sc.textFile("BDA/input/stations-Ostergotland.csv")
lines_precipitation = precipitation_file.map(lambda line: line.split(";"))
lines_stations = stations_file.map(lambda line: line.split(";"))
precReadingsRow = lines_precipitation.map(lambda p: (p[0], p[1], int(p[1].split("-")[0]),
int(p[1].split("-")[1]), p[2], float(p[3]), p[4] ))
stationReadingsRow = lines_stations.map(lambda x: (x[0], x[1]))

stationReadingsString = ["station", "name"]
precReadingsString = ["station", "date", "year", "month", "time", "prec", "quality"]
# Apply the schema to the RDD.
schemaStationReadings = sqlContext.createDataFrame(stationReadingsRow,
stationReadingsString)
schemaPrecReadings = sqlContext.createDataFrame(precReadingsRow,
precReadingsString)
```

wilan057
chrvo878
2020-05-15

```
# Register the DataFrame as a table.
schemaStationReadings.registerTempTable("stationReadingsTable")
schemaPrecReadings.registerTempTable("precReadingsTable")
# Filtering all the stations in OstergOtland
prec_ogotland = schemaPrecReadings.join(schemaStationReadings, 'station')
prec_month = prec_ogotland.groupBy(['station', 'year',
'month']).agg(F.sum('prec').alias('prec'))
prec_month_avg = prec_month.groupBy(['year',
'month']).agg(F.avg('prec').alias('avgMonthlyPrec'))
prec_month_avg.orderBy(['year', 'month'],
ascending=[0,0]).rdd.saveAsTextFile("BDA/output/prec_month_avg_ogotland")
```