Appendix

1a

```
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("/user/x_krisi/data/temperature-readings.csv")
parts = rdd.map(lambda x: x.split(";"))
temp = parts.map(lambda x: Row(station = x[0], date = x[1],
        year = x[1].split("-")[0], time=x[2], value=float(x[3]),
    quality=x[4]))
schemaTemp = sqlContext.createDataFrame(temp)
schemaTemp.registerTempTable("temp")

schemaTemp = schemaTemp.filter(schemaTemp['year'] >= 1950)
schemaTemp = schemaTemp.filter(schemaTemp['year'] <= 2014)


min_temps = schemaTemp.groupBy('year').agg(
    F.min('value').alias(
    'value'))

min_temps = min_temps.join(schemaTemp, ['year', 'value']).select(
    'year', 'station', 'value').orderBy(
    F.desc("value"))

max_temps = schemaTemp.groupBy('year').agg(
    F.max('value').alias(
    'value'))

max_temps = max_temps.join(schemaTemp, ['year', 'value']).select(
        'year', 'station', 'value').orderBy(
        F.desc("value"))


min_temps.rdd.saveAsTextFile("lab2_res/min_temps")
max_temps.rdd.saveAsTextFile("lab2_res/max_temps")
```

2 api

```
schemaTemp.registerTempTable("temp")
schemaTemp = schemaTemp.filter(schemaTemp['year'] >= 1950)
schemaTemp = schemaTemp.filter(schemaTemp['year'] <= 2014)
schemaTemp = schemaTemp.filter(schemaTemp['value'] >= 10)

count_temps = schemaTemp.groupBy(
         'year','month').agg(
         F.count('value').alias(
         'value')).orderBy(
         F.desc('value'))
count_distinct = schemaTemp.select(
     'station','year','month').distinct().groupBy(
     "year","month").count().orderBy(F.desc('count'))

count_temps.rdd.saveAsTextFile("lab2_res/count_temps")
count_distinct.rdd.saveAsTextFile("lab2_res/count_distinct")
```

2 regular sql

```
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("/user/x_krisi/data/temperature-readings.csv")
parts = rdd.map(lambda x: x.split(";"))

temp = parts.map(lambda x: Row(station = x[0], date = x[1],
        year = x[1].split("-")[0],month = x[1].split("-")[1],
        time=x[2], value=float(x[3]),
        quality=x[4]))

schemaTemp = sqlContext.createDataFrame(temp)
schemaTemp.registerTempTable("temp")

count_temps = sqlContext.sql("SELECT␣year,␣month,␣count(value)␣as␣value␣FROM␣temp
␣␣␣␣where␣year>=1950␣and␣year<=2014␣and␣value>=10.0
␣␣␣␣group␣by␣year,␣month␣ORDER␣BY␣value␣DESC")

count_distinct = sqlContext.sql("SELECT␣year,␣month,␣count(DISTINCT␣station)␣as␣count␣FROM
␣␣␣␣temp␣where␣year>=1950␣and␣year<=2014␣and␣value>=10.0
␣␣␣␣group␣by␣year,␣month␣ORDER␣BY␣count␣DESC")

count_temps.rdd.saveAsTextFile("lab2_res/count_temps_sql")
count_distinct.rdd.saveAsTextFile("lab2_res/count_distinct_sql")
```

3

```
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("/user/x_krisi/data/temperature-readings.csv")
parts = rdd.map(lambda x: x.split(";"))
temp = parts.map(lambda x: Row(station = x[0], date = x[1],
          year = x[1].split("-")[0], month = x[1].split("-")[1],
          day = x[1].split("-")[2],time=x[2], value=float(x[3]),
          quality=x[4]))
schemaTemp = sqlContext.createDataFrame(temp)
schemaTemp.registerTempTable("temp")

schemaTemp = schemaTemp.filter(schemaTemp['year'] >= 1960)
schemaTemp = schemaTemp.filter(schemaTemp['year'] <= 2016)

avg_temp = schemaTemp.groupBy(
          'year','month','day','station').agg(
          (F.max('value')+F.min('value')/ 2).alias('value') )

avg_temp = avg_temp.groupBy(
          'year', 'month','station').agg(
           F.avg('value').alias('avg')).orderBy(F.desc('avg'))

avg_temp.rdd.saveAsTextFile("lab2_res/avg_temp")
```

4

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


sc = SparkContext()
sqlContext = SQLContext(sc)

temp_rdd = sc.textFile("/user/x_krisi/data/temperature-readings.csv")
parts = temp_rdd.map(lambda x: x.split(";"))
temp = parts.map(lambda x: Row(station = x[0], value=float(x[3])))
schemaTemp = sqlContext.createDataFrame(temp)
schemaTemp.registerTempTable("temp")

precipitation_rdd = sc.textFile("/user/x_krisi/data/precipitation-readings.csv")
parts = precipitation_rdd.map(lambda x: x.split(";"))
precipitation = parts.map(lambda x: Row(station = x[0],
                                year = x[1].split("-")[0], month = x[1].split("-")[1],
                                day = x[1].split("-")[2],value=float(x[3])))
schemaRain = sqlContext.createDataFrame(precipitation)
schemaRain.registerTempTable("precipitation")

temp_filter = schemaTemp.groupBy('station').agg(F.max('value').alias('max_temp'))
temp_filter = temp_filter.filter(temp_filter['max_temp'] >= 25)
temp_filter = temp_filter.filter(temp_filter['max_temp'] <= 30)


rain_filter = schemaRain.groupBy('station','year','month','day').agg(
        F.sum('value').alias('value'))
rain_filter = rain_filter.groupBy('station').agg(F.max('value').alias('value'))
rain_filter = rain_filter.filter(rain_filter['value'] >= 100)
rain_filter = rain_filter.filter(rain_filter['value'] <= 200)

joined_filter = temp_filter.join(
        rain_filter ,['station']).select(
        'station','max_temp','value').orderBy(F.desc('station')).show()
```

5

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


sc = SparkContext()
sqlContext = SQLContext(sc)

station_rdd = sc.textFile("/user/x_krisi/data/stations-Ostergotland.csv")
parts = station_rdd.map(lambda x: x.split(";"))
stations = parts.map(lambda x: Row(station = x[0]))
schemaStation = sqlContext.createDataFrame(stations)
schemaStation.registerTempTable("stations")
length = schemaStation.count()

precipitation_rdd = sc.textFile("/user/x_krisi/data/precipitation-readings.csv")
parts = precipitation_rdd.map(lambda x: x.split(";"))
precipitation = parts.map(lambda x: Row(station = x[0],
                                    year = x[1].split("-")[0], month = x[1].split("-")[1],
                                    day = x[1].split("-")[2],value=float(x[3])))
schemaRain = sqlContext.createDataFrame(precipitation)
schemaRain.registerTempTable("precipitation")

rain_in_ost = schemaRain.filter(schemaRain['year'] >= 1993)
rain_in_ost = rain_in_ost.filter(rain_in_ost['year'] <= 2016)

rain_in_ost = schemaRain.join(
        schemaStation,['station']).select('year', 'month','value')

rain_in_ost = rain_in_ost.groupBy(
        'year','month').agg((F.sum('value')/length).alias('value')).orderBy(F.desc('value'))

rain_in_ost.rdd.saveAsTextFile('lab2_res/rain_in_ost')
```

6

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

sc = SparkContext()
sqlContext = SQLContext(sc)

station_rdd = sc.textFile("/user/x_krisi/data/stations-Ostergotland.csv")
parts = station_rdd.map(lambda x: x.split(";"))
stations = parts.map(lambda x: Row(station = x[0]))
schemaStation = sqlContext.createDataFrame(stations)
schemaStation.registerTempTable("stations")
length = schemaStation.count()

rdd = sc.textFile("/user/x_krisi/data/temperature-readings.csv")
parts = rdd.map(lambda x: x.split(";"))
temp = parts.map(lambda x: Row(station = x[0], date = x[1],
                                    year = x[1].split("-")[0], month = x[1].split("-")[1],
                                    day = x[1].split("-")[2], value=float(x[3])))
schemaTemp = sqlContext.createDataFrame(temp)
schemaTemp.registerTempTable("temp")

schemaTemp = schemaTemp.join(schemaStation,['station']).select(
     'year','month','value','day','station')

# Long-term avg
long_temp = schemaTemp.filter(schemaTemp['year'] >= 1950)
long_temp = long_temp.filter(long_temp['year'] <= 1980)


long_temp = long_temp.groupBy(
                              'year','month','day','station').agg(
                              (F.max('value')+F.min('value')/ 2).alias('value'))

long_temp = long_temp.groupBy('year', 'month').agg(
           F.avg('value').alias('value'))

long_temp = long_temp.groupBy('month').agg(F.avg('value').alias('value'))

# Avg temp
avg_temp = schemaTemp.filter(schemaTemp['year'] >= 1950)
avg_temp = avg_temp.filter(avg_temp['year'] <= 2014)

avg_temp = avg_temp.groupBy(
                              'year','month','day','station').agg(
                              (F.max('value')+F.min('value')/2).alias('value'))

avg_temp = avg_temp.groupBy(
          'year', 'month').agg(F.avg('value').alias('avg'))

# Difference
joint = avg_temp.join(long_temp, ['month'])
joint = joint.withColumn('diff',joint['avg']-joint['value'])
joint = joint.select(
          'month','year','diff').orderBy(
          F.desc('year'),F.desc('month'))

joint.rdd.repartition(1).saveAsTextFile('lab2_res/joint')
```