Here are the columns that are present in the datasets:

Building.csv – BuildingID, BuildingMgr, BuildingAge, HVACproduct,Country
HVAC.csv – Date, Time, TargetTemp, ActualTemp, System, SystemAge, BuildingID
Now, let's perform analysis on the HVAC dataset to obtain the temperature changes in the building. We are performing this analysis using Spark SQL. The following is the code for performing this analysis.

```
val data = sc.textFile("/home/kiran/Documents/SensorFiles/HVAC.csv")

val header = data.first()

val data1 = data.filter(row => row != header)

case class
hvac_cls(Date:String,Time:String,TargetTemp:Int,ActualTemp:Int,System:Int,S
ystemAge:Int,BuildingId:Int)

val hvac = data1.map(x=>x.split(",")).map(x =>
hvac_cls(x(0),x(1),x(2).toInt,x(3).toInt,x(4).toInt,x(5).toInt,x(6).toInt))
.toDF

hvac.registerTempTable("HVAC")

val hvac1 = sqlContext.sql("select *,IF((targettemp - actualtemp) > 5, '1',
IF((targettemp - actualtemp) < -5, '1', 0)) AS tempchange from HVAC")

hvac1.registerTempTable("HVAC1")
```

- The first three lines of code will remove the header from the CSV file.
- In the 4th line, we are writing a case class holding the schema of the dataset.
- In the 5th line, we are splitting each row of the dataset with the delimiter 'as' and we are mapping the columns to our case class and finally, we are converting it into a data frame.
- In the 6th line, we are creating a table **HVAC** for our dataframe.

- In the 7th line, we are performing an SQL query on the table, which creates one new column **tempchange**, which will set to **1** if there is a temperature change of either +5 or -5 between the actual_temperature and the target_temperature.
- In the 8th line, we are registering that table as **HVAC1.**

Now, let's create a table for the building dataset.

```
val data2 = sc.textFile("/home/kiran/Documents/SensorFiles/building.csv")

val header1 = data2.first()

val data3 = data2.filter(row => row != header1)

case class
building(buildid:Int,buildmgr:String,buildAge:Int,hvacproduct:String,Countr
y:String)

val build = data3.map(x=> x.split(",")).map(x =>
building(x(0).toInt,x(1),x(2).toInt,x(3),x(4))).toDF

build.registerTempTable("building")
```

- The first 3 lines of the dataset will remove the header from the dataset.
- In the 4th line, we have defined a case class holding the schema of the building dataset.
- In the 5th line, we are mapping the dataset to the case class which we have built.
- In the 6th line, we are registering the build dataframe as a table **building.**

Now, we will join both the tables **building and hvac1** as shown below.

```
val build1 = sqlContext.sql("select h.*, b.country, b.hvacproduct from
building b join hvac1 h on buildid = buildingid")

val test = build1.map(x => (new Integer(x(7).toString),x(8).toString))

val test1 = test.filter(x=> {if(x._1==1) true else false})

val test2 = test1.map(x=>(x._2,1)).reduceByKey(_+_).map(item =>
item.swap).sortByKey(false).collect
```

- In the 1st line, we are joining the two datasets using the buildingId.
- In the 2nd line, we are taking the **tempchange** column and the **country** column, which are required to find the maximum temperature change areas.
- In the 3rd line, we are filtering the rows which have a change in temperature, which is identified by **1.**
- In the 4th line, we are taking the country and we are adding **1** to know how many times the temperature in that building has changed. We are applying reduceByKey operation on the data to count the number of times temperature has been changed and finally, we are sorting the in descending order and printing it out.

Below are the results based on our analysis.

```
(473,Finland), (251,France), (248,Hong Kong), (243,Indonesia),
(243,Turkey), (241,China), (237,South Africa), (236,Egypt), (233,Saudi
Arabia), (232,Israel), (232,Canada), (230,Argentina), (230,Singapore),
(228,Mexico), (226,Brazil), (225,Australia), (213,USA), (199,Belgium),
(196,Germany)
```

From the above output, we can say that temperature in **Finland** is changing more frequently followed by **France and Hong Kong.**

If there is a continuous stream of data collected from the sensors, we can automate this analysis using Spark Streaming to know the temperature changes in the real-time, so that we can take accurate measures to reduce the temperature changes.