Case Study-III Sensor Data

The link to download the dataset in below:

https://drive.google.com/drive/folders/1npD2CQrLK44Yg1jxlLeF7EEpE9BzVNtV

building.csv

HVAC.csv

In the above link there are two datasets;
building.csv contains the details of the top 20 buildings all over the world
and
HVAC.csv contains the target temperature and the actual temperature along with the building Id.

HVAC (heating, ventilating/ventilation, and air conditioning) is the technology of indoor and vehicular environmental comfort. Its goal is to provide thermal comfort and acceptable indoor air quality. Through the HVAC sensors, we will get the temperature of the buildings.

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

Objective-1

1. Load HVAC.csv file into temporary table.
2. Add a new column, tempchange – set to 1, if there is a change of greater than +/-5 between actual and target temperature

Solution:
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

- Below code will remove the header from the CSV file.

```
scala> val data = sc.textFile("/sensor/HVAC.csv")
data: org.apache.spark.rdd.RDD[String] = /sensor/HVAC.csv MapPartitionsRDD[3] at textFile at <console>:36

scala> val header = data.first()
header: String = "Date  Time    TargetTemp      ActualTemp      System  SystemAge       BuildingID

scala> val data1 = data.filter(row => row != header)
data1: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[4] at filter at <console>:40
```

- Then next we are writing a case class holding the schema of the dataset.

```
scala> case class hvac_cls(Date:String,Time:String,TargetTemp:Int,ActualTemp:Int,System:Int,SystemAge:Int,BuildingId:Int)
defined class hvac_cls
```

- Then in below code ,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

```
scala> 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).toI
nt)).toDF
hvac: org.apache.spark.sql.DataFrame = [Date: string, Time: string ... 5 more fields]
```

- Excepted output of objective 1 part A : we are creating a table HVAC for our dataframe.

```
scala> hvac.registerTempTable("HVAC")
warning: there was one deprecation warning; re-run with -deprecation for details

scala>
```

And checked if table exists.

Expected Output:

```
scala> sqlContext.sql("show tables").show()
+--------+---------+-----------+
|database|tableName|isTemporary|
+--------+---------+-----------+
| default|      abc|      false|
| default|  college|      false|
|        |     hvac|       true|
+--------+---------+-----------+


scala> val abc = sqlContext.sql("select * from hvac")
abc: org.apache.spark.sql.DataFrame = [Date: string, Time: string ... 5 more fie
lds]
```

```
+--------+--------+----------+----------+------+---------+----------+
|    Date|    Time|TargetTemp|ActualTemp|System|SystemAge|BuildingId|
+--------+--------+----------+----------+------+---------+----------+
|  6/1/13| 0:00:01|        66|        58|    13|       20|         4|
|  6/2/13| 1:00:01|        69|        68|     3|       20|        17|
|  6/3/13| 2:00:01|        70|        73|    17|       20|        18|
|  6/4/13| 3:00:01|        67|        63|     2|       23|        15|
|  6/5/13| 4:00:01|        68|        74|    16|        9|         3|
|  6/6/13| 5:00:01|        67|        56|    13|       28|         4|
|  6/7/13| 6:00:01|        70|        58|    12|       24|         2|
|  6/8/13| 7:00:01|        70|        73|    20|       26|        16|
|  6/9/13| 8:00:01|        66|        69|    16|        9|         9|
| 6/10/13| 9:00:01|        65|        57|     6|        5|        12|
| 6/11/13|10:00:01|        67|        70|    10|       17|        15|
| 6/12/13|11:00:01|        69|        62|     2|       11|         7|
| 6/13/13|12:00:01|        69|        73|    14|        2|        15|
| 6/14/13|13:00:01|        65|        61|     3|        2|         6|
| 6/15/13|14:00:01|        67|        59|    19|       22|        20|
| 6/16/13|15:00:01|        65|        56|    19|       11|         8|
| 6/17/13|16:00:01|        67|        57|    15|        7|         6|
| 6/18/13|17:00:01|        66|        57|    12|        5|        13|
| 6/19/13|18:00:01|        69|        58|     8|       22|         4|
| 6/20/13|19:00:01|        67|        55|    17|        5|         7|
+--------+--------+----------+----------+------+---------+----------+
only showing top 20 rows
```

For Part-B

We are performing an SQL query on the table, which creates one new column **tempchange,** which will set to **1.**

If there is a change in temperature change of either +5 or -5 between the actual temperature and the target temperature.

We are registering that table as **HVAC1**

```
scala> val hvac1 = sqlContext.sql("select *,IF((targettemp - actualtemp) > 5, '1
', IF((targettemp - actualtemp) < -5, '1', 0)) AS tempchange from hvac")
hvac1: org.apache.spark.sql.DataFrame = [Date: string, Time: string ... 6 more f
ields]

scala> hvac1.registerTempTable("HVAC1")
warning: there was one deprecation warning; re-run with -deprecation for details

scala> sqlContext.sql("show tables").show()
+--------+---------+-----------+
|database|tableName|isTemporary|
+--------+---------+-----------+
| default|      abc|      false|
| default|  college|      false|
|        |     hvac|       true|
|        |    hvac1|       true|
```

We have added one column “tempChange”

Expected Output

```
scala> val Temp = sqlContext.sql("select * from hvac1")
Temp: org.apache.spark.sql.DataFrame = [Date: string, Time: string ... 6 more fields]

scala> Temp.show()
+--------+--------+----------+----------+------+---------+----------+----------+
|    Date|    Time|TargetTemp|ActualTemp|System|SystemAge|BuildingId|tempchange|
+--------+--------+----------+----------+------+---------+----------+----------+
| 6/1/13| 0:00:01|        66|        58|    13|       20|         4|         1|
| 6/2/13| 1:00:01|        69|        68|     3|       20|        17|         0|
| 6/3/13| 2:00:01|        70|        73|    17|       20|        18|         0|
| 6/4/13| 3:00:01|        67|        63|     2|       23|        15|         0|
| 6/5/13| 4:00:01|        68|        74|    16|        9|         3|         1|
| 6/6/13| 5:00:01|        67|        56|    13|       28|         4|         1|
| 6/7/13| 6:00:01|        70|        58|    12|       24|         2|         1|
| 6/8/13| 7:00:01|        70|        73|    20|       26|        16|         0|
| 6/9/13| 8:00:01|        66|        69|    16|        9|         9|         0|
|6/10/13| 9:00:01|        65|        57|     6|        5|        12|         1|
|6/11/13|10:00:01|        67|        70|    10|       17|        15|         0|
|6/12/13|11:00:01|        69|        62|     2|       11|         7|         1|
|6/13/13|12:00:01|        69|        73|    14|        2|        15|         0|
|6/14/13|13:00:01|        65|        61|     3|        2|         6|         0|
|6/15/13|14:00:01|        67|        59|    19|       22|        20|         1|
|6/16/13|15:00:01|        65|        56|    19|       11|         8|         1|
|6/17/13|16:00:01|        67|        57|    15|        7|         6|         1|
|6/18/13|17:00:01|        66|        57|    12|        5|        13|         1|
|6/19/13|18:00:01|        69|        58|     8|       22|         4|         1|
|6/20/13|19:00:01|        67|        55|    17|        5|         7|         1|
+--------+--------+----------+----------+------+---------+----------+----------+
only showing top 20 rows

scala>
```

Objective-2

a. Load building.csv file into temporary table

```
scala> val data2 = sc.textFile("/sensor/building.csv")
data2: org.apache.spark.rdd.RDD[String] = /sensor/building.csv MapPartitionsRDD[27] at textFile at <console>:44

scala> val header1 = data2.first()
header1: String = BuildingID,BuildingMgr,BuildingAge,HVACproduct,Country

scala> val data3 = data2.filter(row => row != header1)
data3: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[28] at filter at <console>:48
```

```
File  Edit  View  Search  Terminal  Help
scala> data3.foreach(println)
1,M1,25,AC1000,USA
2,M2,27,FN39TG,France
3,M3,28,JDNS77,Brazil
4,M4,17,GG1919,Finland
5,M5,3,ACMAX22,Hong Kong
6,M6,9,AC1000,Singapore
7,M7,13,FN39TG,South Africa
8,M8,25,JDNS77,Australia
9,M9,11,GG1919,Mexico
10,M10,23,ACMAX22,China
11,M11,14,AC1000,Belgium
12,M12,26,FN39TG,Finland
13,M13,25,JDNS77,Saudi Arabia
14,M14,17,GG1919,Germany
15,M15,19,ACMAX22,Israel
16,M16,23,AC1000,Turkey
17,M17,11,FN39TG,Egypt
18,M18,25,JDNS77,Indonesia
19,M19,14,GG1919,Canada
20,M20,19,ACMAX22,Argentina

scala> case class building(buildid:Int,buildmgr:String,buildAge:Int,hvacproduct:String,Country:String)
defined class building

scala> val build = data3.map(x=> x.split(",")).map(x => building(x(0).toInt,x(1),x(2).toInt,x(3),x(4))).toDF
build: org.apache.spark.sql.DataFrame = [buildid: int, buildmgr: string ... 3 more fields]

scala> build.registerTempTable("building")
warning: there was one deprecation warning; re-run with -deprecation for details

scala> sqlContext.sql("show tables").show()
+--------+---------+-----------+
|database|tableName|isTemporary|
+--------+---------+-----------+
| default|      abc|      false|
| default|  college|      false|
|        | building|       true|
|        |     hvac|       true|
```

Objective- 3

Figure out the number of times, temperature has changes by 5 degrees are more for each country:

a. Join both the tables

We have joined hvac1 and building table on buildingID

```
scala> val build1 = sqlContext.sql("select h.*, b.country, b.hvacproduct from building b join hvac1 h on buildid = buildingid
")
build1: org.apache.spark.sql.DataFrame = [Date: string, Time: string ... 8 more fields]

scala> build1.show()
+-------+--------+----------+----------+------+---------+----------+----------+-------+-----------+
|   Date|    Time|TargetTemp|ActualTemp|System|SystemAge|BuildingId|tempchange|country|hvacproduct|
+-------+--------+----------+----------+------+---------+----------+----------+-------+-----------+
|6/10/13| 9:00:01|        65|        57|     6|        5|        12|         1|Finland|     FN39TG|
|6/18/13|23:13:19|        66|        75|     1|       13|        12|         1|Finland|     FN39TG|
| 6/2/13|13:43:51|        65|        72|    20|       26|        12|         1|Finland|     FN39TG|
|6/13/13| 0:13:20|        67|        77|     8|       19|        12|         1|Finland|     FN39TG|
|6/16/13| 3:13:20|        67|        55|    11|       16|        12|         1|Finland|     FN39TG|
|6/30/13|17:13:20|        65|        57|    17|        9|        12|         1|Finland|     FN39TG|
| 6/1/13|18:13:20|        68|        65|     7|       21|        12|         0|Finland|     FN39TG|
|6/25/13|18:33:07|        70|        66|    20|       20|        12|         0|Finland|     FN39TG|
|6/17/13|16:00:01|        69|        68|    16|        4|        12|         0|Finland|     FN39TG|
| 6/5/13|16:43:51|        69|        69|    19|       15|        12|         0|Finland|     FN39TG|
|6/23/13|10:13:20|        65|        61|     1|        1|        12|         0|Finland|     FN39TG|
|6/29/13|16:13:20|        67|        80|    12|        8|        12|         1|Finland|     FN39TG|
| 6/4/13|21:13:20|        66|        72|     7|        1|        12|         1|Finland|     FN39TG|
| 6/3/13| 2:00:01|        69|        72|     7|       21|        12|         0|Finland|     FN39TG|
|6/16/13|15:00:01|        67|        77|     4|       22|        12|         1|Finland|     FN39TG|
|6/22/13|21:00:01|        70|        77|    13|       12|        12|         1|Finland|     FN39TG|
|6/26/13| 7:43:51|        65|        62|     6|        6|        12|         0|Finland|     FN39TG|
|6/26/13|13:13:20|        65|        63|    20|        9|        12|         0|Finland|     FN39TG|
|6/30/13|17:13:20|        66|        62|    14|       26|        12|         0|Finland|     FN39TG|
|6/10/13| 3:33:07|        70|        78|     5|        9|        12|         1|Finland|     FN39TG|
+-------+--------+----------+----------+------+---------+----------+----------+-------+-----------+
only showing top 20 rows
```

b. Select tempChange and country column

```
scala> val test = build1.map(x => (new Integer(x(7).toString),x(8).toString))
test: org.apache.spark.sql.Dataset[(Integer, String)] = [_1: int, _2: string]

scala> test.show()
+---+-------+
| _1|     _2|
+---+-------+
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  0|Finland|
|  0|Finland|
|  0|Finland|
|  0|Finland|
|  0|Finland|
|  1|Finland|
|  1|Finland|
|  0|Finland|
|  1|Finland|
|  1|Finland|
|  0|Finland|
|  0|Finland|
|  0|Finland|
|  1|Finland|
+---+-------+
only showing top 20 rows
```

c. Filter the rows where tempchange is 1 and count the number of occurrence for each country

- We are filtering the rows which have a change in temperature, which is identified by 1.

```
scala> val test1 = test.filter(x=> {if(x._1==1) true else false})
test1: org.apache.spark.sql.Dataset[(Integer, String)] = [_1: int, _2: string]

scala> test1.show()
+---+-------+
| _1|     _2|
+---+-------+
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
|  1|Finland|
+---+-------+
only showing top 20 rows
```

- 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.

```
scala> val test2=test1.groupBy("_2")
test2: org.apache.spark.sql.RelationalGroupedDataset = org.apache.spark.sql.RelationalGroupedDataset@e926fce

scala> test2.count()
res30: org.apache.spark.sql.DataFrame = [_2: string, count: bigint]

scala> test2.count().withColumnRenamed("_2","country").show()
+------------+-----+
|     country|count|
+------------+-----+
|   Singapore|   63|
|      Turkey|   72|
|     Germany|   48|
|      France|   70|
|   Argentina|   48|
|     Belgium|   51|
|     Finland|  120|
|       China|   69|
|   Hong Kong|   68|
|      Israel|   67|
|         USA|   47|
|      Mexico|   57|
|   Indonesia|   65|
|Saudi Arabia|   61|
|      Canada|   57|
|      Brazil|   67|
|   Australia|   46|
|       Egypt|   56|
|South Africa|   51|
+------------+-----+
```