

Case Study 3 Sensor

For this data analysis, you can download the necessary dataset from this link.

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

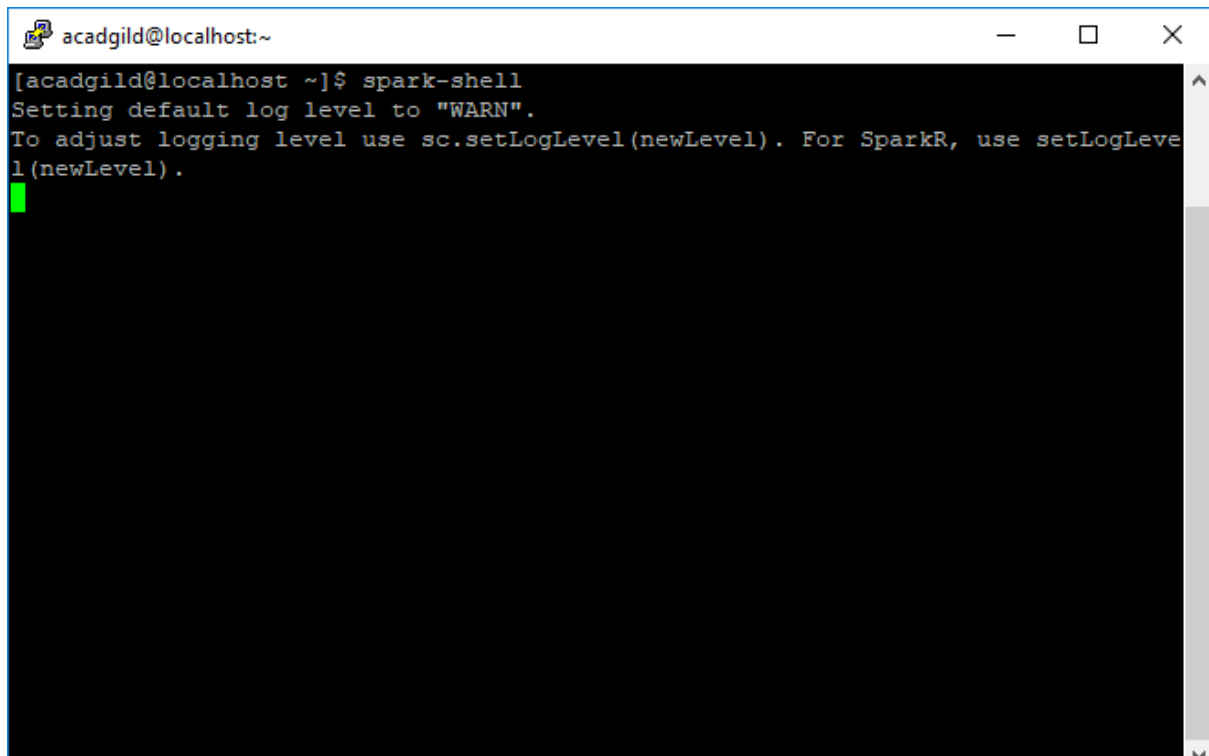
HVAC.csv – Date, Time, TargetTemp, ActualTemp, System, SystemAge, BuildingID

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

Step 1:

Start Spark –shell

A terminal window titled 'acadgild@localhost:~' with standard window controls. The terminal shows the command 'spark-shell' being executed, which sets the default log level to 'WARN' and provides instructions on how to adjust logging levels using 'sc.setLogLevel' or 'setLogLevel' for SparkR. A green cursor is visible on the line following the instructions.

```
acadgild@localhost:~  
[acadgild@localhost ~]$ spark-shell  
Setting default log level to "WARN".  
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).  
^
```

Step 2:

//Load data into spark

```
val sqlContext = new org.apache.spark.sql.SQLContext(sc)
```

```
val data = sc.textFile("/user/acadgild/HVAC.csv")
```

//Remove Header

```
val header = data.first()
```

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

//Define Case Class HVAC

```
case class
```

```
HVAC(Date:String,Time:String,TargetTemp:Int,ActualTemp:Int,System:String,  
SystemAge:String,BuildingID:String)
```

//Convert to DataFrame

```
val dataDF =
```

```
data1.map(_._split(",")).map(h=>HVAC(h(0),h(1),h(2).toInt,h(3).toInt,h(4),h(5),  
h(6))).toDF
```

```
acadgild@localhost:~  
scala> val sqlContext = new org.apache.spark.sql.SQLContext(sc)  
warning: there was one deprecation warning; re-run with -deprecation for details  
sqlContext: org.apache.spark.sql.SQLContext = org.apache.spark.sql.SQLContext@3e0d722f  
  
scala> val data = sc.textFile("/user/acadgild/HVAC.csv")  
data: org.apache.spark.rdd.RDD[String] = /user/acadgild/HVAC.csv MapPartitionsRDD[9] at textFile at <console>:24  
  
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[10] at filter at <console>:28  
  
scala> case class HVAC(Date:String,Time:String,TargetTemp:Int,ActualTemp:Int,System:String,SystemAge:String,BuildingID:String)  
defined class HVAC  
  
scala> val dataDF = data1.map(_.split(",")).map(h => HVAC(h(0),h(1),h(2).toInt,h(3).toInt,h(4),h(5),h(6))).toDF  
dataDF: org.apache.spark.sql.DataFrame = [Date: string, Time: string ... 5 more fields]  
scala> █
```

Step 3:

//Adding new column NewTemp to find difference between TargetTemp and ActualTemp

```
val temp = udf((TargetTemp:Int,ActualTemp:Int) =>  
(TargetTemp,ActualTemp)match  
{  
  case(TargetTemp,ActualTemp) => {TargetTemp-ActualTemp}  
})  
spark.udf.register("NewTemp",temp)  
  
val tempRDD  
=dataDF.withColumn("NewTemp",temp(dataDF.col("TargetTemp"),dataDF.col("ActualTemp"))).toDF
```

```

acagild@localhost:~$ scala
scala> val temp = udf((TargetTemp: Int, ActualTemp: Int) => (TargetTemp, ActualTemp) match {
  | {
  |   case (TargetTemp, ActualTemp) => (TargetTemp - ActualTemp)
  | })
temp: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function2>, IntegerType, Some(List(IntegerType, IntegerType)))

scala>
scala> spark.udf.register("NewTemp", temp)
res4: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function2>, IntegerType, Some(List(IntegerType, IntegerType)))

scala> val tempRDD = dataDF.withColumn("NewTemp", temp(dataDF.col("TargetTemp"), dataDF.col("ActualTemp"))).toDF
tempRDD: org.apache.spark.sql.DataFrame = [Date: string, Time: string ... 6 more fields]

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

```

Step 4:

//Adding New Column tempchange - set to 1, if there is a change of greater than +/-5 between actual and target temperature

```
val temp1 = udf((NewTemp: Int) =>
```

```
if(NewTemp > 5 ) {1} else if(NewTemp > -5 ) {1} else {0}
```

```
}
```

```
val output = tempRDD.withColumn("TempChange", temp1(col("NewTemp")))
```

```
output.show()
```

```

acagild@localhost:~$ scala
scala> val temp1 = udf((NewTemp: Int) =>
  | if(NewTemp > 5 ) {1} else if(NewTemp > -5 ) {1} else {0}
  | )
temp1: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function1>, IntegerType, Some(List(IntegerType)))

scala>
scala> val output = tempRDD.withColumn("TempChange", temp1(col("NewTemp")))

```

```
acadgild@localhost:~  
  
scala> val output = tempRDD.withColumn("TempChange",templ(col("NewTemp")))  
output: org.apache.spark.sql.DataFrame = [Date: string, Time: string ... 7 more fields]  
  
scala> output.show(30)  
  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| Date| Time|TargetTemp|ActualTemp|System|SystemAge|BuildingID|NewTemp|TempChange|  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 6/1/13| 0:00:01| 66| 58| 13| 20| 4| 8| 1|  
| 6/2/13| 1:00:01| 69| 68| 3| 20| 17| 1| 0|  
| 6/3/13| 2:00:01| 70| 73| 17| 20| 18| -3| 0|  
| 6/4/13| 3:00:01| 67| 63| 2| 23| 15| 4| 0|  
| 6/5/13| 4:00:01| 68| 74| 16| 9| 3| -6| 1|  
| 6/6/13| 5:00:01| 67| 56| 13| 28| 4| 11| 1|  
| 6/7/13| 6:00:01| 70| 58| 12| 24| 2| 12| 1|  
| 6/8/13| 7:00:01| 70| 73| 20| 26| 16| -3| 0|  
| 6/9/13| 8:00:01| 66| 69| 16| 9| 9| -3| 0|  
| 6/10/13| 9:00:01| 65| 57| 6| 5| 12| 8| 1|  
| 6/11/13| 10:00:01| 67| 70| 10| 17| 15| -3| 0|  
| 6/12/13| 11:00:01| 69| 62| 2| 11| 7| 7| 1|  
| 6/13/13| 12:00:01| 69| 73| 14| 2| 15| -4| 0|  
| 6/14/13| 13:00:01| 65| 61| 3| 2| 6| 4| 0|  
| 6/15/13| 14:00:01| 67| 59| 19| 22| 20| 8| 1|  
| 6/16/13| 15:00:01| 65| 56| 19| 11| 8| 9| 1|  
| 6/17/13| 16:00:01| 67| 57| 15| 7| 6| 10| 1|  
| 6/18/13| 17:00:01| 66| 57| 12| 5| 13| 9| 1|  
| 6/19/13| 18:00:01| 69| 58| 8| 22| 4| 11| 1|  
| 6/20/13| 19:00:01| 67| 55| 17| 5| 7| 12| 1|  
| 6/21/13| 20:00:01| 69| 72| 7| 5| 17| -3| 0|  
| 6/22/13| 21:00:01| 66| 69| 6| 29| 9| -3| 0|  
| 6/23/13| 22:00:01| 67| 65| 6| 18| 20| 2| 0|  
| 6/24/13| 23:00:01| 67| 61| 15| 13| 6| 6| 1|  
| 6/25/13| 0:13:19| 70| 71| 19| 14| 18| -1| 0|  
| 6/26/13| 1:13:19| 67| 71| 1| 6| 8| -4| 0|  
| 6/27/13| 2:13:19| 69| 68| 16| 14| 14| 1| 0|  
| 6/28/13| 3:13:19| 65| 56| 8| 18| 15| 9| 1|  
| 6/29/13| 4:13:19| 66| 77| 7| 6| 13| -11| 1|  
| 6/30/13| 5:13:19| 70| 63| 7| 13| 5| 7| 1|  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
only showing top 30 rows  
  
scala> █
```

Step 5:

//Loading HVAC.csv file into temporary table

output.registerTempTable("HVACTable")

```
acadgild@localhost:~  
  
scala> output.registerTempTable("HVACTable")  
warning: there was one deprecation warning; re-run with -deprecation for details  
  
scala> █
```

Step 6:

val result1 = sqlContext.sql("select * from HVACTable")

result1.show()

```
scala> val result1 = sqlContext.sql("select * from HVACTable")
result1: org.apache.spark.sql.DataFrame = [Date: string, Time: string ... 7 more fields]

scala> result1.show()
+-----+-----+-----+-----+-----+-----+-----+-----+
| Date|    Time|TargetTemp|ActualTemp|System|SystemAge|BuildingID|NewTemp|TempChange|
+-----+-----+-----+-----+-----+-----+-----+-----+
| 6/1/13| 0:00:01|      66|      58|    13|     20|        4|      8|        1|
| 6/2/13| 1:00:01|      69|      68|     3|     20|       17|     11|        0|
| 6/3/13| 2:00:01|      70|      73|    17|     20|       18|    -3|        0|
| 6/4/13| 3:00:01|      67|      63|     2|     23|       15|     4|        0|
| 6/5/13| 4:00:01|      68|      74|    16|     9|        3|    -6|        1|
| 6/6/13| 5:00:01|      67|      56|    13|    28|        4|    11|        1|
| 6/7/13| 6:00:01|      70|      58|    12|    24|        2|    12|        1|
| 6/8/13| 7:00:01|      70|      73|    20|    26|       16|    -3|        0|
| 6/9/13| 8:00:01|      66|      69|    16|     9|        9|    -3|        0|
| 6/10/13| 9:00:01|      65|      57|     6|     5|       12|     8|        1|
| 6/11/13|10:00:01|      67|      70|    10|    17|       15|    -3|        0|
| 6/12/13|11:00:01|      69|      62|     2|    11|        7|     7|        1|
| 6/13/13|12:00:01|      69|      73|    14|     2|       15|    -4|        0|
| 6/14/13|13:00:01|      65|      61|     3|     2|        6|     4|        0|
| 6/15/13|14:00:01|      67|      59|    19|    22|       20|     8|        1|
| 6/16/13|15:00:01|      65|      56|    19|    11|        8|     9|        1|
| 6/17/13|16:00:01|      67|      57|    15|     7|        6|    10|        1|
| 6/18/13|17:00:01|      66|      57|    12|     5|       13|     9|        1|
| 6/19/13|18:00:01|      69|      58|     8|    22|        4|    11|        1|
| 6/20/13|19:00:01|      67|      55|    17|     5|        7|    12|        1|
+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 20 rows

scala> █
```

Objective 2

Load building.csv file into temporary table

Step 1 :

//Load data

val sqlContext = new org.apache.spark.sql.SQLContext(sc)

val data = sc.textFile("/user/acadgild/building.csv")

//Remove Header

val header = data.first()

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

//Define Building Case class

case class

building(BuildingID:String,BuildingMgr:String,BuildingAge:String,HVACproduct:String,Country:String)

//convert to dataframe

```
val dataDF =  
data1.map(_._split(",")).map(b=>building(b(0),b(1),b(2),b(3),b(4))).toDF
```

```
acadgild@localhost:~  
scala> val sqlContext = new org.apache.spark.sql.SQLContext(sc)  
warning: there was one deprecation warning; re-run with -deprecation for details  
sqlContext: org.apache.spark.sql.SQLContext = org.apache.spark.sql.SQLContext@16de04ae  
  
scala> val data = sc.textFile("/user/acadgild/building.csv")  
data: org.apache.spark.rdd.RDD[String] = /user/acadgild/building.csv MapPartitionsRDD[259] at textFile at <console>:24  
  
scala> val header = data.first()  
header: String = BuildingID,BuildingMgr,BuildingAge,HVACproduct,Country  
  
scala> val data1 = data.filter(row => row != header)  
data1: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[260] at filter at <console>:28  
  
scala> case class building(BuildingID:String,BuildingMgr:String,BuildingAge:String,HVACproduct:String,Country:String)  
defined class building  
  
scala> val dataDF = data1.map(_._split(",")).map(b =>building(b(0),b(1),b(2),b(3),b(4))).toDF  
dataDF: org.apache.spark.sql.DataFrame = [BuildingID: string, BuildingMgr: string ... 3 more fields]  
  
scala> █
```

Step 2:

```
//Loading building.csv file into temporary table  
  
dataDF.registerTempTable("buildingTable")
```

```
acadgild@localhost:~  
scala> //Loading building.csv file into temporary table  
  
scala> dataDF.registerTempTable("buildingTable")  
warning: there was one deprecation warning; re-run with -deprecation for details  
  
scala> █
```

Step 3:

```
val result2 = sqlContext.sql("select * from buildingTable")  
  
result2.show()
```

acadgild@localhost:~

```
scala> val result2 = sqlContext.sql("select * from buildingTable")
result2: org.apache.spark.sql.DataFrame = [BuildingID: string, BuildingMgr: string ... 3 more fields]
```

```
scala> result2.show()
```

BuildingID	BuildingMgr	BuildingAge	HVACproduct	Country
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>
```

Objective 3:

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

1) Join both the tables.

```
val joinTable1 = sqlContext.sql("select * from HVACTable h join buildingTable b on h.BuildingID = b.BuildingID")
```

```
joinTable1.show()
```



```
scala> val joinTable1 = sqlContext.sql("select * from HVACTable h join buildingTable b on h.BuildingID = b.BuildingID")
joinTable1: org.apache.spark.sql.DataFrame = [Date: string, Time: string ... 12 more fields]

scala> joinTable1.show()
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Date| Time|TargetTemp|ActualTemp|System|SystemAge|BuildingID|NewTemp|TempChange|BuildingID|BuildingMgr|BuildingAge|HVACproduct| Country|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 6/12/13|11:00:01| 69| 62| 2| 11| 7| 7| 1| 7| M7| 13| FN39TG|South Africa|
| 6/20/13|19:00:01| 67| 55| 17| 5| 7| 12| 1| 7| M7| 13| FN39TG|South Africa|
| 6/27/13| 8:43:51| 68| 61| 15| 28| 7| 7| 1| 7| M7| 13| FN39TG|South Africa|
| 6/1/13|12:43:51| 69| 79| 6| 15| 7| -10| 1| 7| M7| 13| FN39TG|South Africa|
| 6/5/13|22:13:20| 67| 73| 18| 19| 7| -6| 1| 7| M7| 13| FN39TG|South Africa|
| 6/13/13| 6:33:07| 66| 78| 9| 11| 7| -12| 1| 7| M7| 13| FN39TG|South Africa|
| 6/1/13| 0:00:01| 69| 60| 5| 8| 7| 9| 1| 7| M7| 13| FN39TG|South Africa|
| 6/4/13| 3:00:01| 69| 71| 13| 28| 7| -2| 0| 7| M7| 13| FN39TG|South Africa|
| 6/12/13|11:00:01| 70| 63| 14| 6| 7| 7| 1| 7| M7| 13| FN39TG|South Africa|
| 6/1/13| 6:13:19| 69| 64| 9| 28| 7| 5| 0| 7| M7| 13| FN39TG|South Africa|
| 6/2/13| 7:13:19| 67| 71| 15| 2| 7| -4| 0| 7| M7| 13| FN39TG|South Africa|
| 6/9/13|14:13:19| 68| 80| 5| 9| 7| -12| 1| 7| M7| 13| FN39TG|South Africa|
| 6/14/13|19:13:19| 69| 56| 3| 25| 7| 13| 1| 7| M7| 13| FN39TG|South Africa|
| 6/18/13|11:33:07| 67| 64| 8| 29| 7| 3| 0| 7| M7| 13| FN39TG|South Africa|
| 6/18/13|23:13:19| 67| 69| 3| 7| 7| -2| 0| 7| M7| 13| FN39TG|South Africa|
| 6/28/13| 9:43:51| 65| 68| 8| 5| 7| -3| 0| 7| M7| 13| FN39TG|South Africa|
| 6/8/13|19:43:51| 70| 74| 12| 3| 7| -4| 0| 7| M7| 13| FN39TG|South Africa|
| 6/18/13|17:45:56| 69| 78| 8| 8| 7| -9| 1| 7| M7| 13| FN39TG|South Africa|
| 6/6/13|17:45:56| 65| 63| 4| 28| 7| 2| 0| 7| M7| 13| FN39TG|South Africa|
| 6/2/13| 0:13:19| 70| 74| 2| 27| 7| -4| 0| 7| M7| 13| FN39TG|South Africa|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 20 rows

scala>
```

2) Select tempchange and country column

```
val joinTable2 = sqlContext.sql("select h.TempChange,b.Country from HVACTable h join buildingTable b on h.BuildingID = b.BuildingID GROUP BY TempChange,Country")
```

```
joinTable2.show()
```

```
scala> val joinTable2 = sqlContext.sql("select h.TempChange,b.Country from HVACTable h join buildingTable b on h.BuildingID = b.BuildingID GROUP BY TempChange,Country")
joinTable2: org.apache.spark.sql.DataFrame = [TempChange: int, Country: string]

scala> joinTable2.show()
+-----+-----+
|TempChange| Country|
+-----+-----+
| 0| Israel|
| 1| Egypt|
| 1| Germany|
| 1|South Africa|
| 0| Australia|
| 0| France|
| 0| Hong Kong|
| 1| Mexico|
| 1| Argentina|
| 0| Canada|
| 1| USA|
| 0| China|
| 1| Australia|
| 0| Brazil|
| 0| Singapore|
| 1| Israel|
| 1| France|
| 0| Belgium|
| 0| USA|
| 1| Turkey|
+-----+-----+
only showing top 20 rows

scala>
```

3) Filter the rows where tempchange is 1 and count the number of occurrence for each country

```
val joinTable3 = sqlContext.sql("select b.Country,Count(h.TempChange) as Count from HVACTable h join buildingTable b on h.BuildingID = b.BuildingID GROUP BY TempChange,Country HAVING TempChange = 1")
```

```
joinTable3.show()
```

```
scala> val joinTable3 = sqlContext.sql("select b.Country,Count(h.TempChange) as Count from HVACTable h join buildingTable b on h.BuildingID = b.BuildingID GROUP BY TempChange,Country HAVING TempChange = 1")
joinTable3: org.apache.spark.sql.DataFrame = [Country: string, Count: bigint]

scala> joinTable3.show()
+-----+-----+
| Country|Count|
+-----+-----+
| Egypt| 236|
| Germany| 196|
| South Africa| 237|
| Mexico| 228|
| Argentina| 230|
| USA| 213|
| Australia| 225|
| Israel| 232|
| France| 251|
| Turkey| 243|
| Brazil| 226|
| Belgium| 199|
| Hong Kong| 248|
| Saudi Arabia| 233|
| Singapore| 230|
| Finland| 473|
| China| 241|
| Indonesia| 243|
| Canada| 232|
+-----+-----+
```