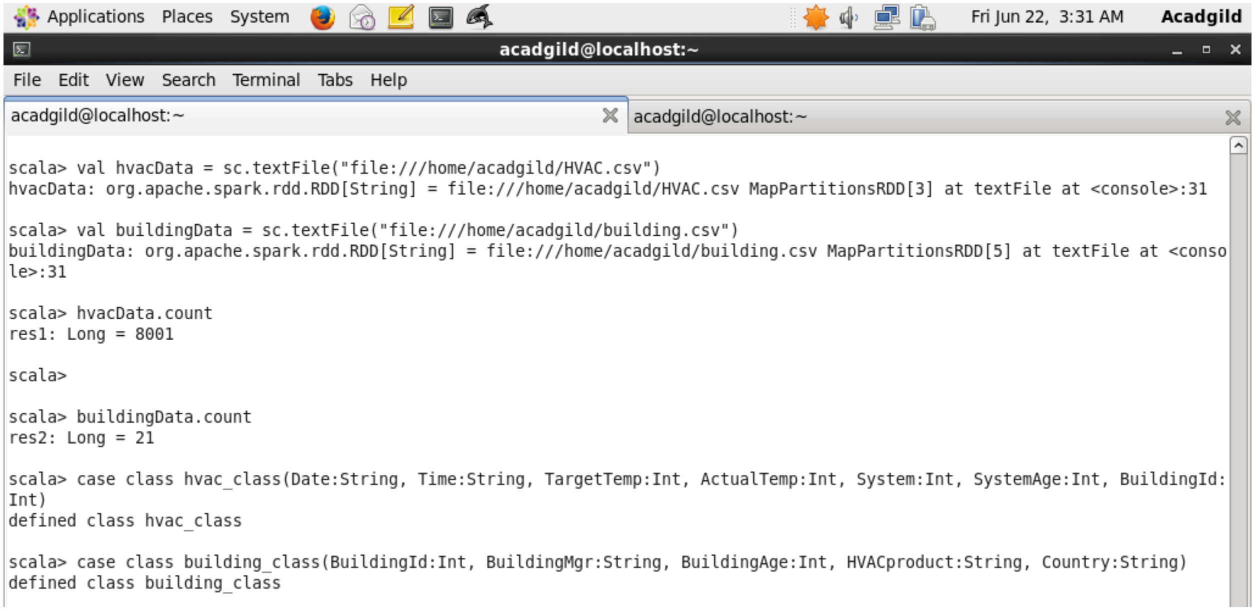
Loading csv files for hvac data and building data

Using count operation to make sure that data is loaded

Defining case classes which will be used later

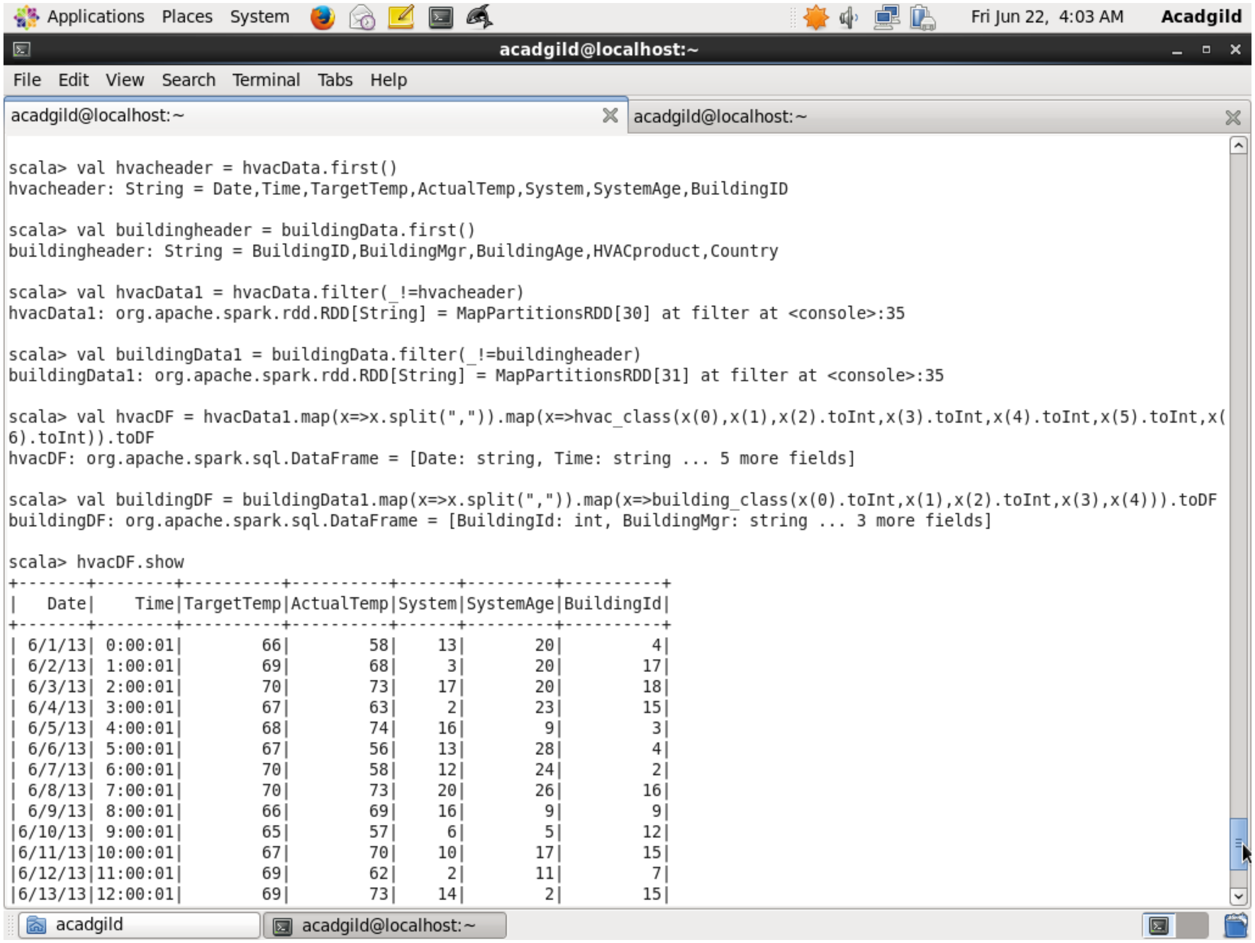


```
scala> val hvacData = sc.textFile("file:///home/acadgild/HVAC.csv")
hvacData: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/HVAC.csv MapPartitionsRDD[3] at textFile at <console>:31

scala> val buildingData = sc.textFile("file:///home/acadgild/building.csv")
buildingData: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/building.csv MapPartitionsRDD[5] at textFile at <conso
le>:31

scala> hvacData.count
res1: Long = 8001

scala>

scala> buildingData.count
res2: Long = 21

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

scala> case class building_class(BuildingId:Int, BuildingMgr:String, BuildingAge:Int, HVACproduct:String, Country:String)
defined class building_class
```

Storing header in a separate variable and then filtering data file accordingly

Creating dataframe using splitting using map and then using case classes defined earlier and finally applying toDF function

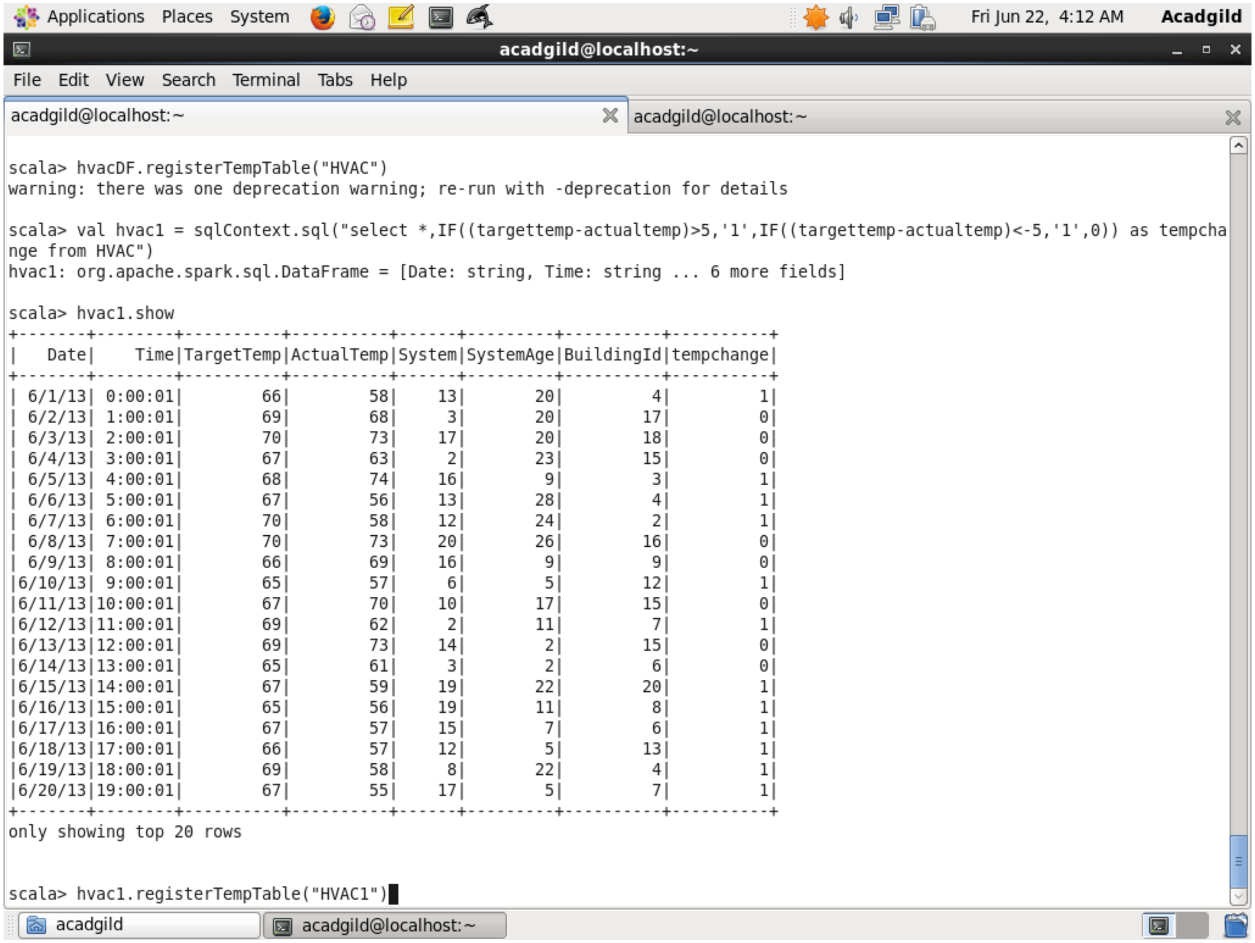
Using show function to see if dataframe was created successsfully



```
scala> val hvacheader = hvacData.first()
hvacheader: String = Date,Time,TargetTemp,ActualTemp,System,SystemAge,BuildingID

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

scala> val hvacData1 = hvacData.filter(_!=hvacheader)
hvacData1: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[30] at filter at <console>:35

scala> val buildingData1 = buildingData.filter(_!=buildingheader)
buildingData1: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[31] at filter at <console>:35

scala> val hvacDF = hvacData1.map(x=>x.split(",")).map(x=>hvac_class(x(0),x(1),x(2).toInt,x(3).toInt,x(4).toInt,x(5).toInt,x(
6).toInt)).toDF
hvacDF: org.apache.spark.sql.DataFrame = [Date: string, Time: string ... 5 more fields]

scala> val buildingDF = buildingData1.map(x=>x.split(",")).map(x=>building_class(x(0).toInt,x(1),x(2).toInt,x(3),x(4))).toDF
buildingDF: org.apache.spark.sql.DataFrame = [BuildingId: int, BuildingMgr: string ... 3 more fields]

scala> hvacDF.show
+-------+--------+----------+----------+------+---------+----------+
|   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|
```

Objective 1 completed. New column added if difference is more than 5 or less than -5

Storing the result in new temp table

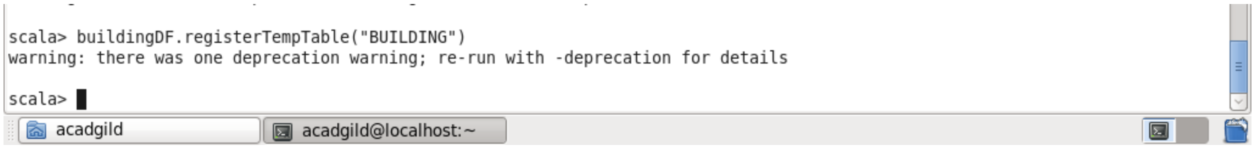


```
scala> hvacDF.registerTempTable("HVAC")
warning: there was one deprecation warning; re-run with -deprecation for details

scala> val hvac1 = sqlContext.sql("select *,IF((targettemp-actualtemp)>5,'1',IF((targettemp-actualtemp)<-5,'1',0)) as tempcha
nge from HVAC")
hvac1: org.apache.spark.sql.DataFrame = [Date: string, Time: string ... 6 more fields]

scala> hvac1.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> hvac1.registerTempTable("HVAC1")
```

Objective 2:

Loading building data in temporary table

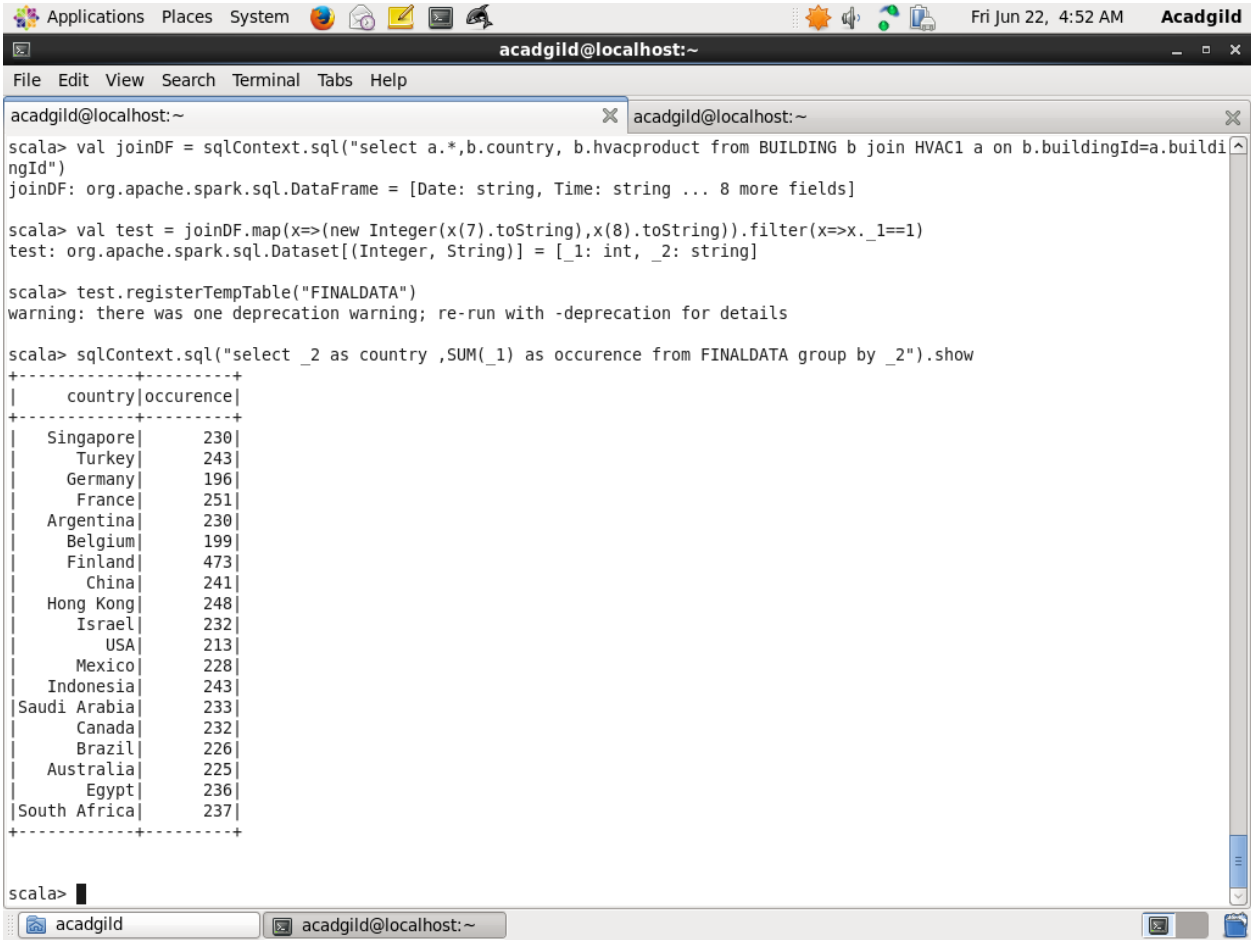
Just using buildingDF dataframe created earlier make temporary table



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

scala> 
```

Objective 3:

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



```
scala> val joinDF = sqlContext.sql("select a.*,b.country, b.hvacproduct from BUILDING b join HVAC1 a on b.buildingId=a.buildi
ngId")
joinDF: org.apache.spark.sql.DataFrame = [Date: string, Time: string ... 8 more fields]

scala> val test = joinDF.map(x=>(new Integer(x(7).toString),x(8).toString)).filter(x=>x._1==1)
test: org.apache.spark.sql.Dataset[(Integer, String)] = [_1: int, _2: string]

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

scala> sqlContext.sql("select _2 as country ,SUM(_1) as occurence from FINALDATA group by _2").show
+------------+---------+
|     country|occurence|
+------------+---------+
|   Singapore|      230|
|      Turkey|      243|
|     Germany|      196|
|      France|      251|
|   Argentina|      230|
|     Belgium|      199|
|     Finland|      473|
|       China|      241|
|   Hong Kong|      248|
|      Israel|      232|
|         USA|      213|
|      Mexico|      228|
|   Indonesia|      243|
|Saudi Arabia|      233|
|      Canada|      232|
|      Brazil|      226|
|   Australia|      225|
|       Egypt|      236|
|South Africa|      237|
+------------+---------+


scala> 
```