Assignment Submission- Case Study 3: Sensor Data

There are two datasets; **building.csv** contains the details of the top 20 buildings all over the world.
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

- Load HVAC.csv file into temporary table.
- Add a new column, tempchange -set to 1, if there is a change of greater than +/-5 between actual and target temperature.
- Load building.csv file into temporary table.
- Figure out the number of times, temperature has changed by 5 degrees or more for each country.

SOLUTION:

Comments are added in the code for each step performed.

```
package SQL

import org.apache.spark.sql.SparkSession

object SparkSQLUseCase1 {

// Creating case/pojo classes for both the sheets of data HVAC and BUILDING

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

  case class
building(buildId:Int,buildMgr:String,buildAge:Int,hvacProduct:String,Country:String
)

  def main(args: Array[String]): Unit = {
```

```
 //Creating spark session object
    val spark = SparkSession.builder()
      .master("local")
      .appName("Sensor Data Use Case")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()

      println("Spark session object created")

    //Set the log level as warning
    spark.sparkContext.setLogLevel("WARN")

    // reading HVAC data from file
    val data =
spark.sparkContext.textFile("E:\\ACADGILD\\SCALA_SPARK\\CaseStudy_SQL\\HVAC.csv")

    println("HVAC Data count-->"+ data.count())

    //Removing header from the data
    val header = data.first()
    val data1 = data.filter(row => row!=header)

    println("Header removed from the data !")

    //For implicit conversions like converting RDDs and sequences  to DataFrames
    import spark.implicits._

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

    //showing hvac data in the dataframe
    hvac.show()

    //creating table view
    hvac.registerTempTable("HVAC")

    println("Dataframe Registered as table !")


    //adding a derived column to above hvac table

    val hvac1 = spark.sql("select *,IF((TargetTemp- ActualTemp)> 5, 1,
IF((TargetTemp- ActualTemp) < -5, 1, 0)) AS tempChange from HVAC")

    hvac1.show

    hvac1.registerTempTable("HVAC1")

    println("Dataframe Registered as table !")

    //reading building data from file
    val data2 =
spark.sparkContext.textFile("E:\\ACADGILD\\SCALA_SPARK\\CaseStudy_SQL\\building.csv
")

    println("Building Data count-->"+ data2.count)

    //Removing header from the data
    val header1 = data2.first()
    val data3 = data2.filter(row => row!=header1)

    println("Header removed from the building data")

    //Now let us create the building dataframe
    val build = data3.map(row => row.split(",")).map(valueArr =>
building(valueArr(0).toInt, valueArr(1), valueArr(2).toInt, valueArr(3),
```

```
valueArr(4)))
      .toDF()

    build.show

    //creating table view
    build.registerTempTable("BUILDING")

    println("Buildings data registered as building table")

    //Joining the 2 tables
    val build1 = spark.sql("select h.*, b.country, b.hvacProduct from BUILDING b
join HVAC1 h on b.buildId = h.BuildingId")

    build1.show()

    // getting country and tempChange columns from above dataframe
    val tempCountry = build1.map( values=> (new Integer(values(7).toString),
values(8).toString))

    tempCountry.show()

    //filtering having ones
    val tempCountryOnes = tempCountry.filter(x => {if(x._1==1) true else false})

    tempCountryOnes.show()

    //getting count for each country

    tempCountryOnes.groupBy("_2").count().show()

  }

}
```

OUTPUT:

```
+------------+-----+
|          _2|count|
+------------+-----+
|   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|
+------------+-----+


Process finished with exit code 0
```