CASE STUDY 3

Objective1: Load HVAC.csv file into temporary table

Sol: Step1: create a baseRDD and load csv file into it

➔ Val baseRDD = sc.textFile("/Dataset1/HVAC.csv")

Step2: Remove the header

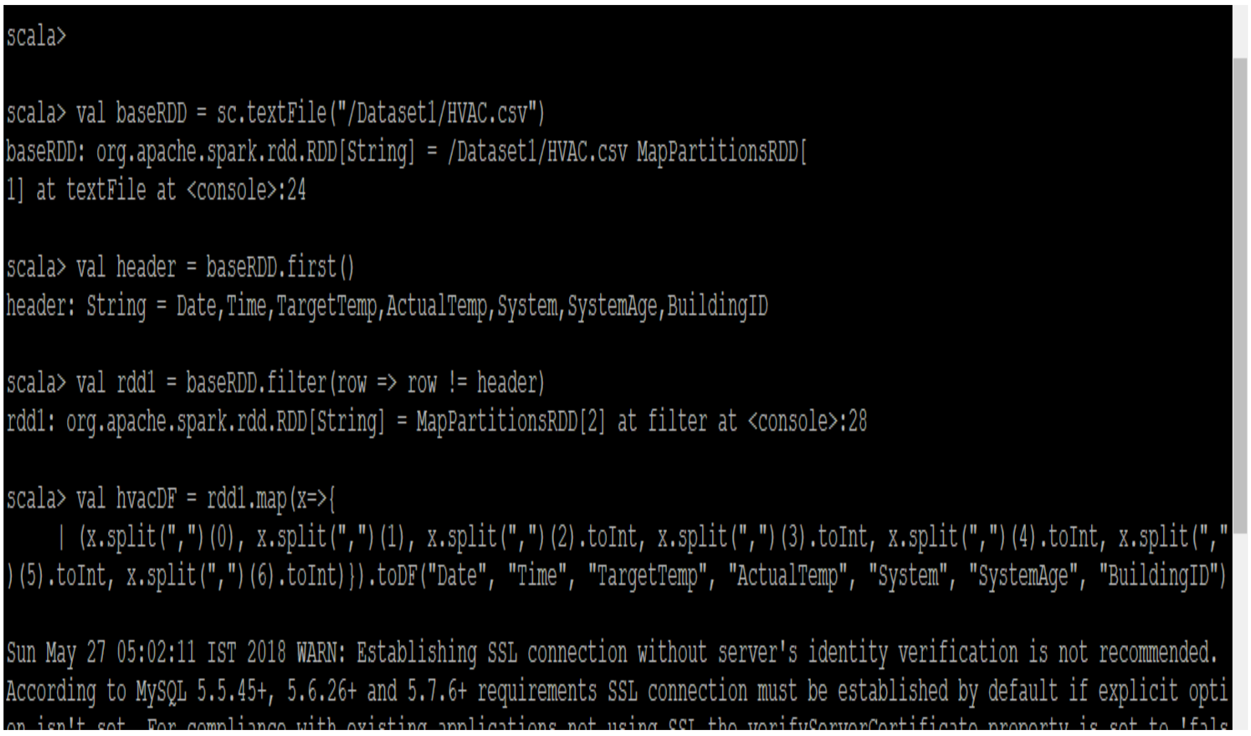
➔ val header = baseRDD.first()
➔ val rdd1 = baseRDD.filter(row => row != header)

Step3: Create dataframe

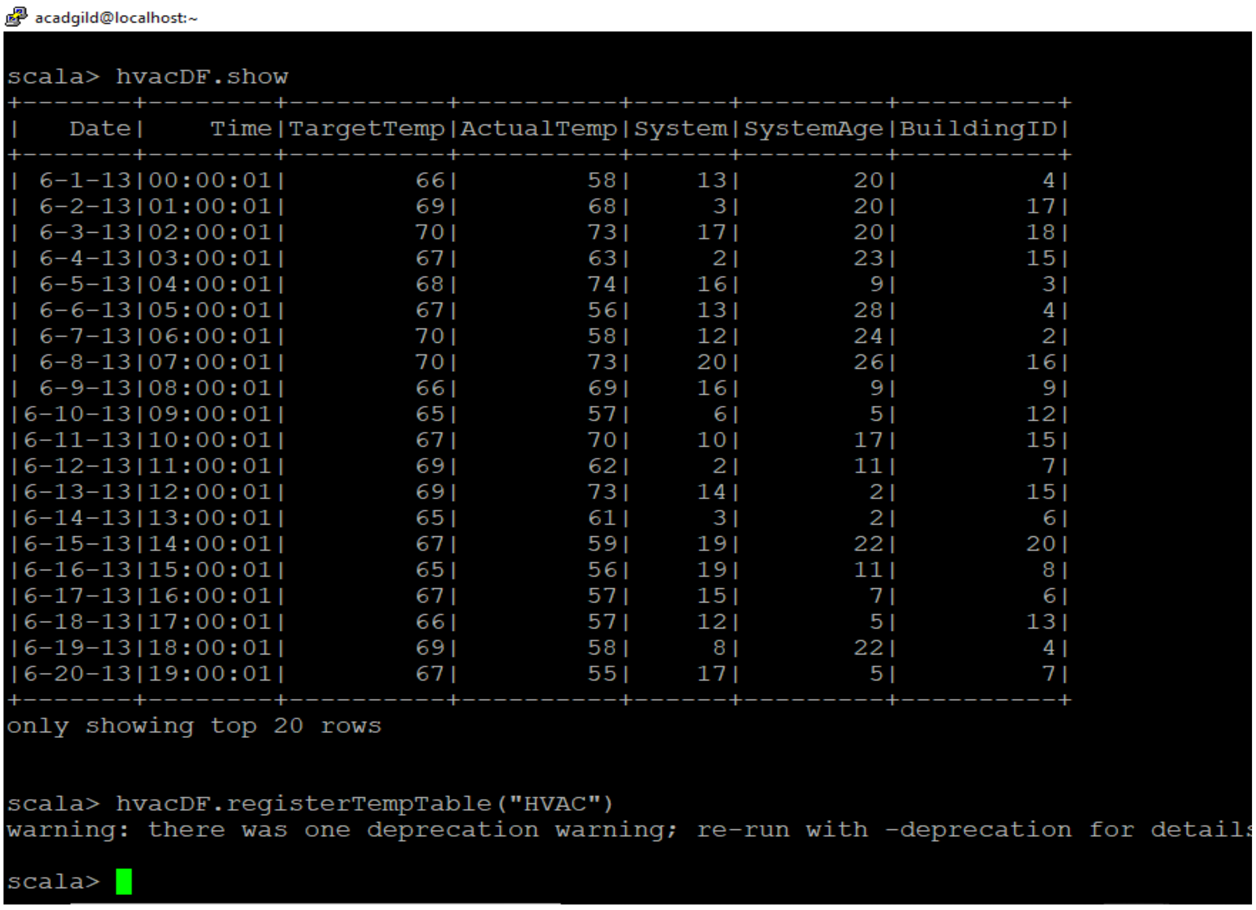
➔ val hvacDF = rdd1.map(x=>{(x.split(",")(0), x.split(",")(1), x.split(",")(2).toInt, x.split(",")(3).toInt, x.split(",")(4).toInt, x.split(",")(5).toInt, x.split(",")(6).toInt)}).toDF("Date", "Time", "TargetTemp", "ActualTemp", "System", "SystemAge", "BuildingID")

Step4: Register as temp table

➔ hvacDF.registerTempTable("HVAC")



```
scala>

scala> val baseRDD = sc.textFile("/Dataset1/HVAC.csv")
baseRDD: org.apache.spark.rdd.RDD[String] = /Dataset1/HVAC.csv MapPartitionsRDD[
1] at textFile at <console>:24

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

scala> val rdd1 = baseRDD.filter(row => row != header)
rdd1: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at filter at <console>:28

scala> val hvacDF = rdd1.map(x=>{
     | (x.split(",")(0), x.split(",")(1), x.split(",")(2).toInt, x.split(",")(3).toInt, x.split(",")(4).toInt, x.split(","
)(5).toInt, x.split(",")(6).toInt)}).toDF("Date", "Time", "TargetTemp", "ActualTemp", "System", "SystemAge", "BuildingID")

Sun May 27 05:02:11 IST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended.
According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit opti
```

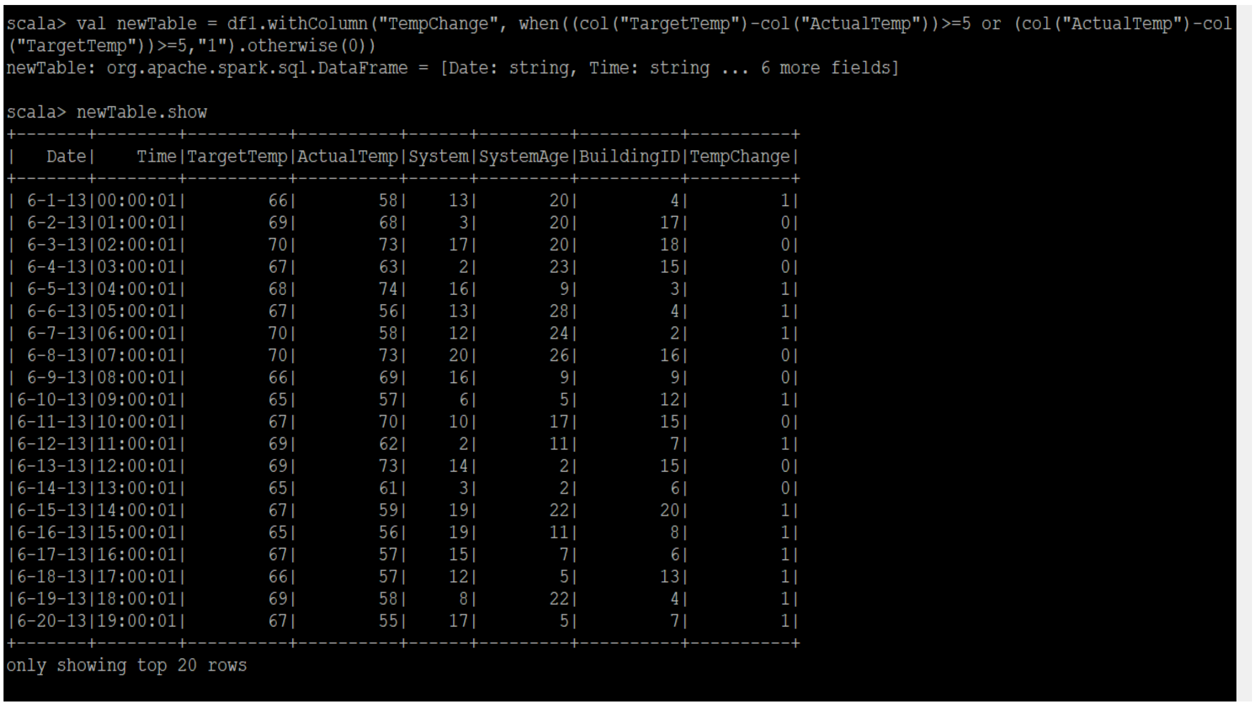

acadgild@localhost:~

```
scala> hvacDF.show
+--------+--------+----------+----------+------+---------+----------+
|    Date|    Time|TargetTemp|ActualTemp|System|SystemAge|BuildingID|
+--------+--------+----------+----------+------+---------+----------+
|  6-1-13|00:00:01|        66|        58|    13|       20|         4|
|  6-2-13|01:00:01|        69|        68|     3|       20|        17|
|  6-3-13|02:00:01|        70|        73|    17|       20|        18|
|  6-4-13|03:00:01|        67|        63|     2|       23|        15|
|  6-5-13|04:00:01|        68|        74|    16|        9|         3|
|  6-6-13|05:00:01|        67|        56|    13|       28|         4|
|  6-7-13|06:00:01|        70|        58|    12|       24|         2|
|  6-8-13|07:00:01|        70|        73|    20|       26|        16|
|  6-9-13|08:00:01|        66|        69|    16|        9|         9|
| 6-10-13|09:00:01|        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


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

scala>
```

Add a new column, tempchange - set to 1, if there is a change of greater than +/-5 between actual and target temperature

Sol: Command Used->> val newTable = df1.withColumn("TempChange", when((col("TargetTemp")-col("ActualTemp"))>=5 or (col("ActualTemp")-col("TargetTemp"))>=5,"1").otherwise(0))

newTable.show



```
scala> val newTable = df1.withColumn("TempChange", when((col("TargetTemp")-col("ActualTemp"))>=5 or (col("ActualTemp")-col
("TargetTemp"))>=5,"1").otherwise(0))
newTable: org.apache.spark.sql.DataFrame = [Date: string, Time: string ... 6 more fields]

scala> newTable.show
+--------+--------+----------+----------+------+---------+----------+----------+
|    Date|    Time|TargetTemp|ActualTemp|System|SystemAge|BuildingID|TempChange|
+--------+--------+----------+----------+------+---------+----------+----------+
|  6-1-13|00:00:01|        66|        58|    13|       20|         4|         1|
|  6-2-13|01:00:01|        69|        68|     3|       20|        17|         0|
|  6-3-13|02:00:01|        70|        73|    17|       20|        18|         0|
|  6-4-13|03:00:01|        67|        63|     2|       23|        15|         0|
|  6-5-13|04:00:01|        68|        74|    16|        9|         3|         1|
|  6-6-13|05:00:01|        67|        56|    13|       28|         4|         1|
|  6-7-13|06:00:01|        70|        58|    12|       24|         2|         1|
|  6-8-13|07:00:01|        70|        73|    20|       26|        16|         0|
|  6-9-13|08:00:01|        66|        69|    16|        9|         9|         0|
| 6-10-13|09:00:01|        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
```

<u>Objective2</u>: Load building.csv file into temporary table

Sol: Step1: create a baseRDD2 and load csv file into it

- ➔ val baseRDD2 = sc.textFile("/Dataset1/building.csv")

Step2: Remove the header

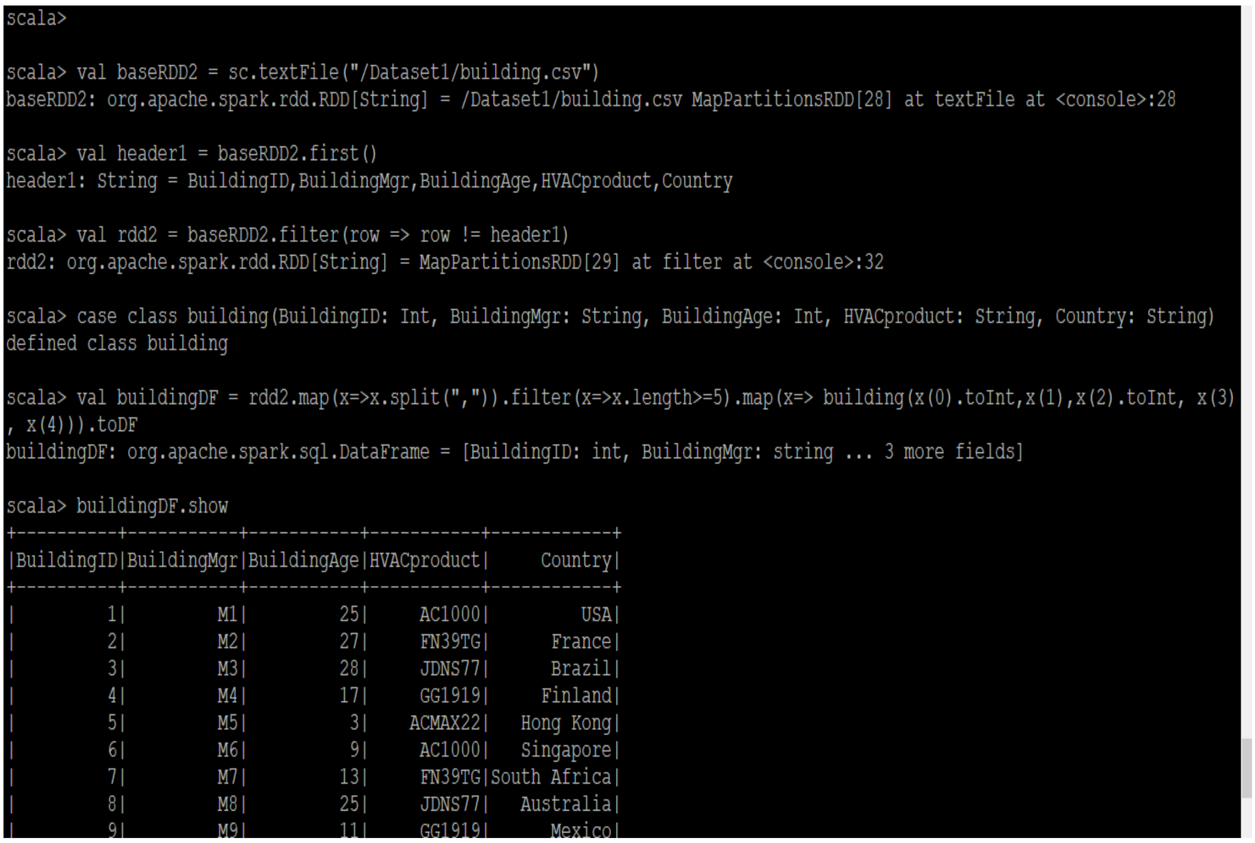
- ➔ val header1 = baseRDD2.first()
- ➔ val rdd2 = baseRDD2.filter(row => row != header1)

Step3: Create dataframe by defining case class

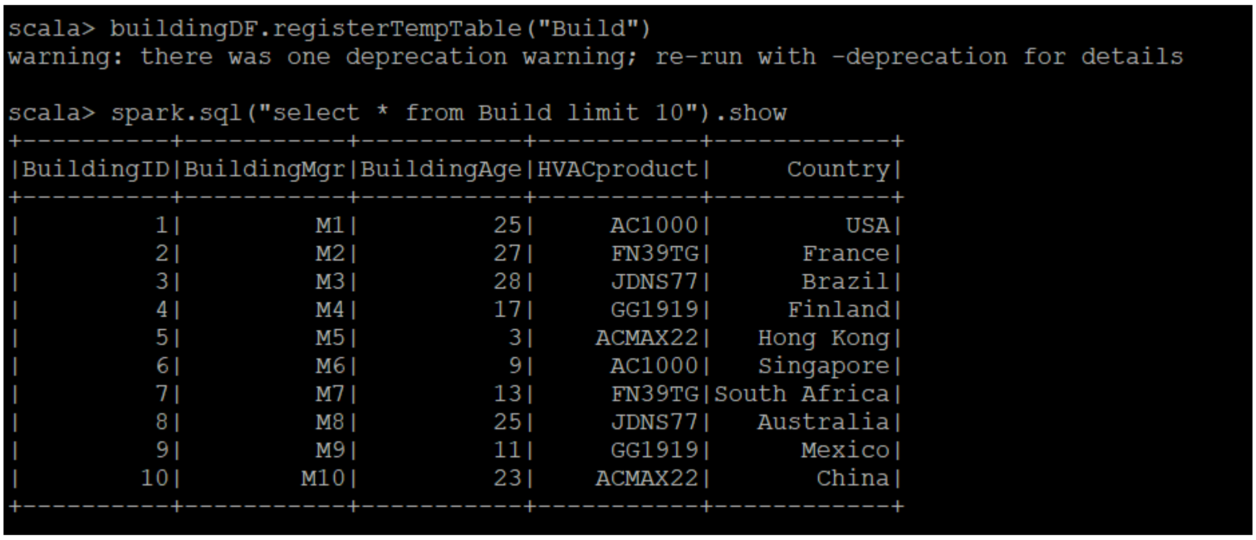
- ➔ case class building(BuildingID: Int, BuildingMgr: String, BuildingAge: Int, HVACproduct: String, Country: String)
- ➔ val buildingDF = rdd2.map(x=>x.split(",")).filter(x=>x.length>=5).map(x=> building(x(0).toInt,x(1),x(2).toInt, x(3), x(4))).toDF

Step4: Register as temp table

- ➔ buildingDF.registerTempTable("Build")



```
scala>

scala> val baseRDD2 = sc.textFile("/Dataset1/building.csv")
baseRDD2: org.apache.spark.rdd.RDD[String] = /Dataset1/building.csv MapPartitionsRDD[28] at textFile at <console>:28

scala> val header1 = baseRDD2.first()
header1: String = BuildingID,BuildingMgr,BuildingAge,HVACproduct,Country

scala> val rdd2 = baseRDD2.filter(row => row != header1)
rdd2: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[29] at filter at <console>:32

scala> case class building(BuildingID: Int, BuildingMgr: String, BuildingAge: Int, HVACproduct: String, Country: String)
defined class building

scala> val buildingDF = rdd2.map(x=>x.split(",")).filter(x=>x.length>=5).map(x=> building(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> buildingDF.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|
```



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

scala> spark.sql("select * from Build limit 10").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|
+----------+-----------+-----------+-----------+------------+
```

<u>Objective3</u>: Figure out the number of times, temperature has changed by 5 degrees or more for each country

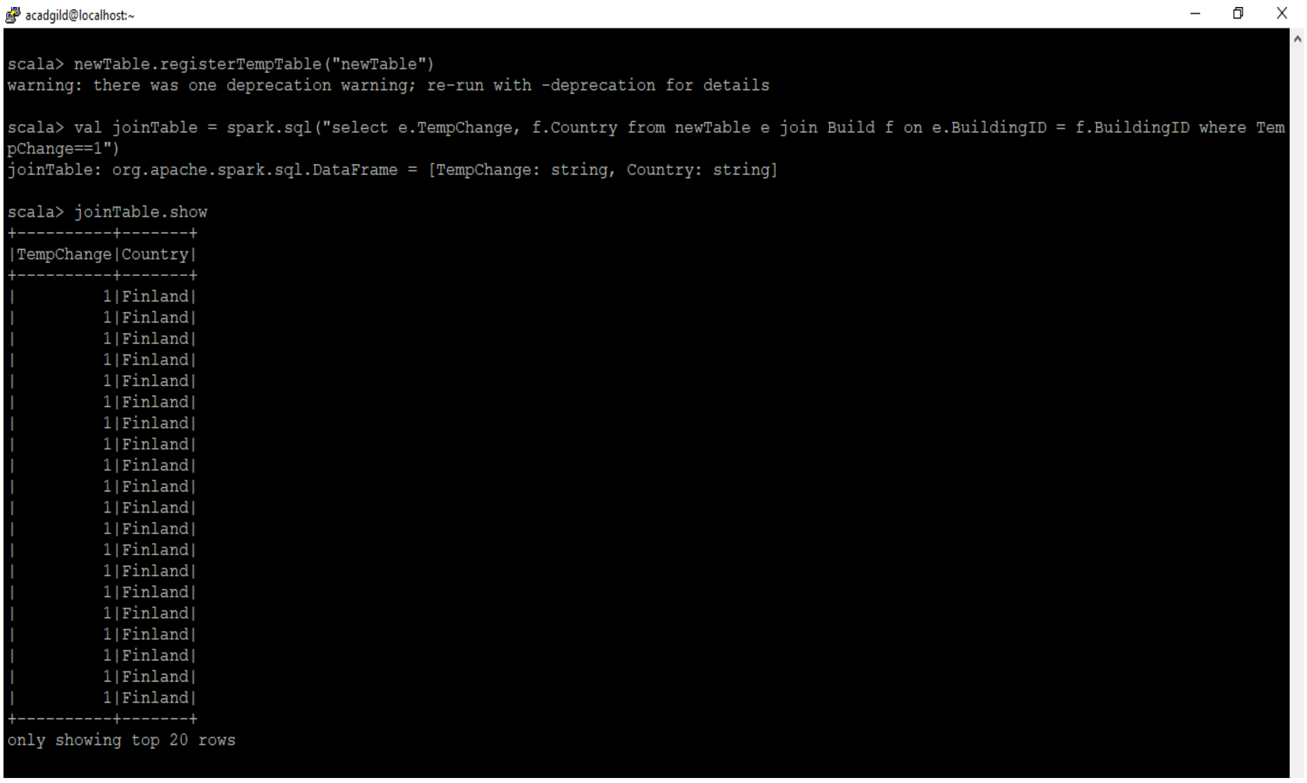
Sol: Step1: Register the newTable(with TempChange column as tempTable)

➔ newTable.registerTempTable("newTable")

Step2: perform join of both tables by showing two country and TempChange==1 column

➔ val joinTable = spark.sql("select e.TempChange, f.Country from newTable e join Build f on e.BuildingID = f.BuildingID where TempChange==1")

➔ joinTable.show



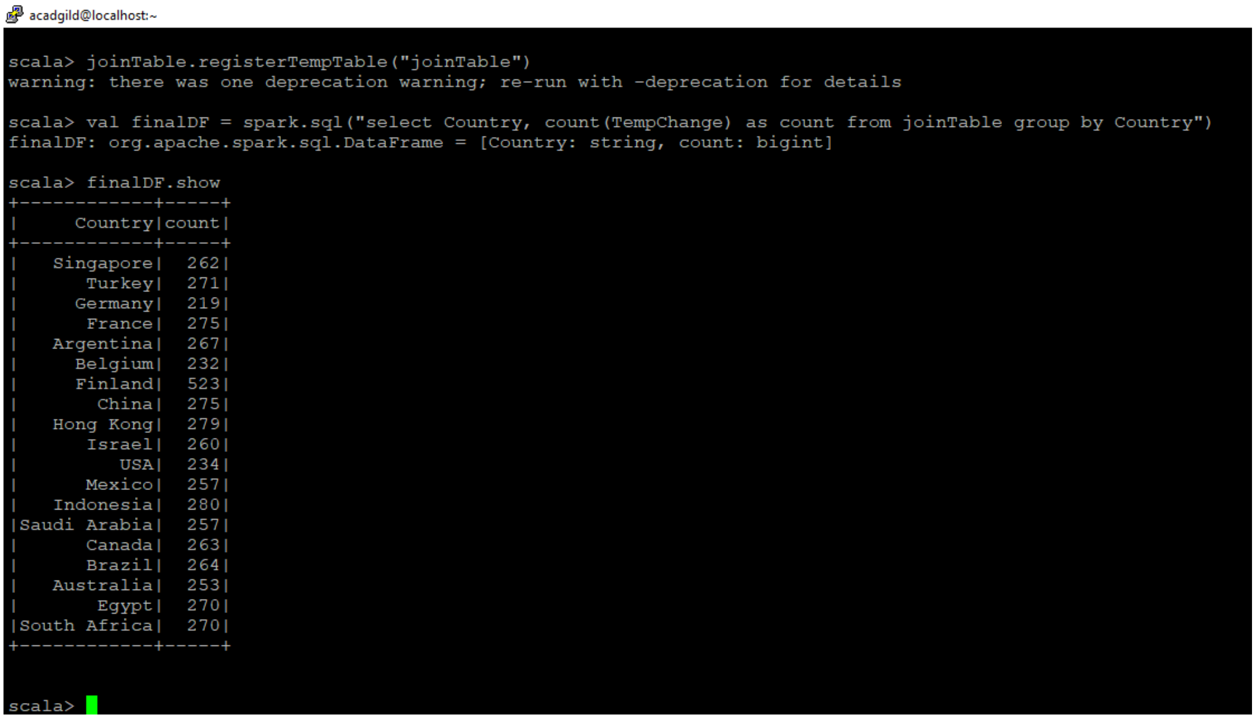
```
scala> newTable.registerTempTable("newTable")
warning: there was one deprecation warning; re-run with -deprecation for details

scala> val joinTable = spark.sql("select e.TempChange, f.Country from newTable e join Build f on e.BuildingID = f.BuildingID where Tem
pChange==1")
joinTable: org.apache.spark.sql.DataFrame = [TempChange: string, Country: string]

scala> joinTable.show
+----------+-------+
|TempChange|Country|
+----------+-------+
|         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
```

Step3: Register the joinTable as tempTable

- ➔ joinTable.registerTempTable("joinTable")

Step4: select the count of TempChange and list corresponding to Country column

- ➔ val finalDF = spark.sql("select Country, count(TempChange) as count from joinTable group by Country")
- ➔ finalDF.show



acadgild@localhost:~

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

scala> val finalDF = spark.sql("select Country, count(TempChange) as count from joinTable group by Country")
finalDF: org.apache.spark.sql.DataFrame = [Country: string, count: bigint]

scala> finalDF.show
+------------+-----+
|     Country|count|
+------------+-----+
|   Singapore|  262|
|      Turkey|  271|
|     Germany|  219|
|      France|  275|
|   Argentina|  267|
|     Belgium|  232|
|     Finland|  523|
|       China|  275|
|   Hong Kong|  279|
|      Israel|  260|
|         USA|  234|
|      Mexico|  257|
|   Indonesia|  280|
|Saudi Arabia|  257|
|      Canada|  263|
|      Brazil|  264|
|   Australia|  253|
|       Egypt|  270|
|South Africa|  270|
+------------+-----+


scala>
```