BIG DATA HADOOP AND SPARK DEVLOPMENT

CASE STUDY III

Table of Contents:

BIG DATA HADOOPAND SPARK DEVELOPMENT

1. Introduction

In this case study, the given tasks are performed and Output of the tasks are recorded in the form of Screenshots.

2. Objective

This case study consolidates the deeper understanding of the Sessions

3. Problem Statement

- Task 1
  - Objective 1
    - 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
  - Objective 2
    - Load building.csv file into temporary table
  - Objective 3
    - Figure out the number of times, temperature has changed by 5 degrees or more for each country:
    - Join both the tables.
    - Select tempchange and country column
    - Filter the rows where tempchange is 1 and count the number of occurrence for each country

4. Expected Output

- Task 1

Working with Sensor Data 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

Objective -2:

Load building.csv file into temporary table

Objective – 3:

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

1. Join both the tables.
2. Select tempchange and country column
3. Filter the rows where tempchange is 1 and count the number of occurrence for each country

Whole code is as follows:

```
import org.apache.spark.sql.SparkSession
object SparkSQLUseCase {

  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 = {
    println("hey scala")
    val spark = SparkSession
      .builder()
      .master("local")
```

```
      .appName("Spark SQL Use Case 1")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()
    println("Spark Session Object created")

    val data = spark.sparkContext
      .textFile("C:\\Users\\king\\Desktop\\HVAC.csv")
      println("HVAC Data->>"+data.count())
      val header = data.first() //extract header
      println("Header is: "+header)
      val data1 = data.filter(row => row != header)
      println("HVAC Data with no Header")
      //For implicit conversions like converting RDDs and sequences to DataFrames
      import spark.implicits._
      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).toInt)).toDF()
      hvac.printSchema()
      hvac.show()
      println("HVAC Dataframe created !")
      hvac.registerTempTable("HVAC")
      println("Dataframe Registered as table !")

    // Now here we are adding one new column to get the temperature range condition

    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("Data Frame Registered as HVAC1 table !")

      val data2 = spark.sparkContext
      .textFile("C:\\Users\\king\\Desktop\\building.csv")
      println("Building Data->>"+data2.count())
      val bheader = data2.first() //extract header
      println("BHeader is: "+bheader)
      val data3 = data2.filter(row => row != bheader)
      println("Building Data with no Header")
      println("Building Data with no header count->>"+data3.count())
      // create data frame for building
      val build = data3.map(x=> x.split(",")).map(x =>
      building(x(0).toInt,x(1),x(2).toInt,x(3),x(4))).toDF
      build.show()

      // Register the table
      build.registerTempTable("BUILDING")
      println("Data Frame Registered as BUILDING table !")
      //Now join the two tables
      val build1 = spark.sql("select h.*, b.country, b.hvacproduct from building b
join hvac1 h on b.buildid = h.buildingid")
      build1.show()
      //Select tempchange and country column
      val tempCountry = build1.map(x => (new Integer(x(7).toString),x(8).toString))
      tempCountry.show()
      //Filter the rows where tempchange is 1 and count number of occurrence for
each country.
      val tempCountryOnes = tempCountry.filter(x=> {if(x._1==1) true else false})
      tempCountryOnes.show()
      tempCountryOnes.groupBy("_2").count.show
       }
}
```

Answer: Now initially we are setting up the SaprkSession to continue for the given case study and then we proceed for data handling as per Objective1. Below screenshot defines the Spark session parameter.

```
def main(args: Array[String]): Unit = {
  println("hey scala")
  val spark = SparkSession
    .builder()
    .master( master = "local")
    .appName( name = "Spark SQL Use Case 1")
    .config("spark.some.config.option", "some-value")
    .getOrCreate()
  println("Spark Session Object created")
```

Now next step is to get HVAC.csv after removing the Header record.

```
val data = spark.sparkContext
  .textFile( path = "C:\\Users\\king\\Desktop\\HVAC.csv")
  println("HVAC Data->>"+data.count())
  val header = data.first() //extract header
  println("Header is: "+header)
  val data1 = data.filter(row => row != header)
  println("HVAC Data with no Header")
```

Objective 1.1: Now we are going to define the Data Frame using the Case class and the define statements and load the data in temporary table.

```
import org.apache.spark.sql.SparkSession
object SparkSQLUseCase {

  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
          )
```

```
//For implicit conversions like converting RDDs and sequences to DataFrames
import spark.implicits._
val hvac = data1.map(x=>x.split( regex = ",")).map(x =>
hvac_cls(x(0),x(1),x(2).toInt,x(3).toInt,x(4).toInt,x(5).toInt,x(6).toInt)).toDF()
hvac.printSchema()
hvac.show()
println("HVAC Dataframe created !")
hvac.registerTempTable( tableName = "HVAC")
println("Dataframe Registered as table !")
```

Output:

```
+-------+--------+----------+----------+------+---------+-----------+
|   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

HVAC Dataframe created !
```

Objective 1.2: Add a new column, tempchange - set to 1, if there is a change of greater than +/-5 between actual and target temperature

```
// Now here we are adding one new column to get the temperature range condition

val hvac1 = spark.sql( sqlText = "select *,IF((targettemp - actualtemp) > 5, '1',IF((targettemp - actualtemp) < -5, '1', 0)) AS tempchange from HVAC")
  hvac1.show()
  hvac1.registerTempTable( tableName = "HVAC1")

  println("Data Frame Registered as HVAC1 table !")
```

Output:

```
+-------+--------+----------+----------+------+---------+-----------+----------+
|   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

Data Frame Registered as HVAC1 table !
```

Objective 2: Load building.csv file into temporary table. Define the case class for the Building table structure as below.

```
import org.apache.spark.sql.SparkSession
object SparkSQLUseCase {

  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
          )
```

```
val data2 = spark.sparkContext
.textFile( path = "C:\\Users\\king\\Desktop\\building.csv")
println("Building Data->>"+data2.count())
val bheader = data2.first() //extract header
println("BHeader is: "+bheader)
val data3 = data2.filter(row => row != bheader)
println("Building Data with no Header")
println("Building Data with no header count->>"+data3.count())
// create data frame for building
val build = data3.map(x=> x.split( regex = ",")).map(x =>
building(x(0).toInt,x(1),x(2).toInt,x(3),x(4))).toDF
build.show()

// Register the table
build.registerTempTable( tableName = "BUILDING")
println("Data Frame Registered as BUILDING table !")
```

Output:

```
+-------+--------+--------+-----------+-------------+
|buildid|buildmgr|buildAge|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|
+-------+--------+--------+-----------+-------------+

Data Frame Registered as BUILDING table !
```

Objective – 3: Figure out the number of times, temperature has changed by 5 degrees or more for each country: 1. Join both the tables. 2. Select tempchange and country column 3. Filter the rows where tempchange is 1 and count the number of occurrence for each country

```
//Filter the rows where tempchange is 1 and count number of occurrence for each country.
val tempCountryOnes = tempCountry.filter(x=> (if(x._1==1) true else false))
tempCountryOnes.show()
tempCountryOnes.groupBy( col1 = "_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|
+------------+-----+
```