Session 22: Interview Preparation-II

Assignment – Case Study III- Machine and Sensor Data Analysis

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

- The first three lines of code will remove the header from the CSV file.
- In the 4th line, we are writing a case class holding the schema of the dataset.
- In the 5th line, we are splitting each row of the dataset with the delimiter 'as' and we are mapping the columns to our case class and finally, we are converting it into a data frame.
- In the 6th line, we are creating a table **HVAC** for our dataframe.
- In the 7th line, we are performing an SQL query on the table, which creates one new column **tempchange**, which will set to **1** if there is a temperature change of either +5 or -5 between the actual_temperature and the target_temperature.
- In the 8th line, we are registering that table as **HVAC1.**

```
scala> val hvacdata = sc.textFile("/usr/HVAC.csv")
hvacdata: org.apache.spark.rdd.RDD[String] = /usr/HVAC.csv MapPartitionsRDD[54] at textFile at <console>:24

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

scala> val actualHvacData= hvacdata.filter(row=>row!=hvacdataHeader)
actualHvacData: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[55] at filter at <console>:28

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

scala> val hvac = actualHvacData.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: org.apache.spark.sql.DataFrame = [Date: string, Time: string ... 5 more fields]

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

scala> val hvacDF = spark.sql("select *,IF((targettemp-actualtemp)>5,'1',IF((targettemp-actualtemp)<-5,'1',0)) AS tempchange from HVACTem
p")
hvacDF: org.apache.spark.sql.DataFrame = [Date: string, Time: string ... 6 more fields]
```

Objective 2

- The first 3 lines of the dataset will remove the header from the dataset.
- In the 4th line, we have defined a case class holding the schema of the building dataset.
- In the 5th line, we are mapping the dataset to the case class which we have built.
- In the 6th line, we are registering the build dataframe as a table **building.**

```
scala> val data = sc.textFile("/usr/building.csv")
data: org.apache.spark.rdd.RDD[String] = /usr/building.csv MapPartitionsRDD[59] 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[60] at filter at <console>:28

scala> case class building(building:Int,buildmgr:String,buildAge:Int,hvacproduct:String,country:String)
defined class building

scala> val build = data1.map(x=>s.split(",")).map(x=>building(x(0).toInt,x(1),x(2).toInt,x(3),x(4))).toDF
<console>:32: error: not found: value s
       val build = data1.map(x=>s.split(",")).map(x=>building(x(0).toInt,x(1),x(2).toInt,x(3),x(4))).toDF
                             ^

scala> val build = data1.map(x=>x.split(",")).map(x=>building(x(0).toInt,x(1),x(2).toInt,x(3),x(4))).toDF
build: org.apache.spark.sql.DataFrame = [building: int, buildmgr: string ... 3 more fields]

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

Objective 3
Figure out the number of times, temperature has changed by 5 degrees or more for each country:

- In the 1st line, we are joining the two datasets using the buildingId.
- In the 2nd line, we are taking the **tempchange** column and the **country** column, which are required to find the maximum temperature change areas.
- In the 3rd line, we are filtering the rows which have a change in temperature, which is identified by **1.**
- In the 4th line, we are taking the country and we are adding **1** to know how many times the temperature in that building has changed. We are applying reduceByKey operation on the data to count the number of times temperature has been changed and finally, we are sorting the in descending order and printing it out.

```
scala> val build = spark.sql("select h.*, b.country,b.hvacproduct from building b join HVAC1 h on buildingID = building")
build: org.apache.spark.sql.DataFrame = [Date: string, Time: string ... 8 more fields]

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

scala> val max_temp_fields_filtered = max_temp_fields.filter(x=>{if(x._1==1) true else false})
max_temp_fields_filtered: org.apache.spark.sql.Dataset[(Integer, String)] = [_1: int, _2: string]

scala> val max_temperature_count = max_temp_fields_filtered.map(x=>(x._2,1)).reduceByKey(_+_).map(item=>item.swap).sortByKey(false).colle
ct
```

Below are the results based on our analysis.

```
scala> (473,Finland), (251,France), (248,Hong Kong), (243,Indonesia), (243,Turkey), (241,China), (237,South Africa), (236,Egypt), (233,Sa
udi Arabia), (232,Israel), (232,Canada), (230,Argentina), (230,Singapore), (228,Mexico), (226,Brazil), (225,Australia), (213,USA), (199,B
elgium), (196,Germany)
```