// Say there is AWS EMR cluster with some Spark job (​ see code below ​ ).

// EMR option ​ maximizeResourceAllocation ​ is enabled.

// It computes amount of purchases for middle-aged customers by each day of month.

// How would you modify the code (​ make it better from performance and style

// standpoints ) ​ ?

// Please describe in details all of your modifications (​ any modified/added line ​ ).

// BEFORE REFACTORING

val numberOfDaysInMonth = 30​// assume always 30 days in month

val getDayOfMonth ​ = ​ udf {... ​ get ​ number of day of month from ​ some date ​ .. ​ }

val transactions ​ = ​ sql​.read​.option​(..).​load​(..).where​ (..)​// transactions for 30 days from DB

val transactionsByDay =

for​ ​(i ​<-​ ​1 to numberOfDaysInMonth​ ) yield​ {

i ->​ transactions​.filter​(getDayOfMonth​(col​("​date​"))===lit(i))

}

val customers ​= ​sql​.read​.option​(..).​load​(..)

.​filter​(!(col​("​age​")​<30​||​ col​("​age​") >​50​))// only middle-aged customers from DB

val customerTransactionsByDay = transactionsByDay​.map( kv=>​ kv match{

case​ (numOfDay​, dataframe​ ) =>

numOfDay->​ {

dataframe​ .join​ (

customers​,

customers​("​customer​")​ ===​ dataframe​("​customer​"),

"left\_outer​ "

)

.​where​(dataframe​ ("​customer​")​!==​ lit(​ null))

.​drop​(dataframe​("​customer​"))

}

case​ \_=> ???

}).​toMap

var​ customerTransactionsAllDays = customerTransactionsByDay​(0)

for​(i <-​ 1 until​ customerTransactionsByDay​.length​ ){

customerTransactionsAllDays =

customerTransactionsAllDays​ .unionAll​(customerTransactionsByDay​(i))

}

customerTransactionsAllDays​ .groupBy(

col​("​customer​"),

getDayOfMonth​(col​("​date​")​).​as​("​day​")

)

.​agg​ (sum​("​amount​").​as​("​amount​"))

.​write​ .option​ (..​).​ save​ (...)​// save to DB