**HIVE**

create table store(store\_id int, store\_num string, city string, address string, open\_dt string, close\_dt string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ",";

create table employee(emp\_id int, emp\_num int, store\_num string, emp\_name string, joining\_dt string, designation string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ",";

create table promotions(promo\_cd\_id int, promo\_cd string, description string, promo\_start\_dt string, promo\_end\_dt string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ",";

create table loyalty(loyalty\_member\_id int, cust\_id int, card\_no string, joining\_dt string, points int) ROW FORMAT DELIMITED FIELDS TERMINATED BY ",";

create table product(product\_id int, product\_cd string, add\_dt string, remove\_dt string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ",";

create table trans\_codes(trans\_code\_id int, trans\_cd string, description string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ",";

**SPARK**

val inputfile = sc.textFile("/input/trans\_log.csv")

val csv = inputfile.map(\_.split(","))

import sqlContext.implicits.\_

import org.apache.spark.sql.Row

import org.apache.spark.sql.types.\_

val impData = csv.filter(x=>(x(1) == "TT" || x(1) == "LL" || x(1) == "PP"))

val TT = impData.filter(x=> (x(1) == "TT")).map(x=> Row(x(0).toInt, x(1), x(2).toInt, x(3), x(4).toDouble, x(5).toDouble, x(6).toInt, x(7), x(8).toInt, x(9).toInt, x(10)))

val PP = impData.filter(x=> (x(1) == "PP")).map(x=> Row(x(0).toInt, x(1), x(2), x(3), x(4).toInt, x(5).toInt, x(6)))

val LL = impData.filter(x=> (x(1) == "LL")).map(x=> Row(x(0).toInt, x(1), x(2), x(3), x(4).toInt, x(5).toInt, x(6)))

val ttSchema = StructType(Seq(StructField("seq\_num", IntegerType, true),

StructField("trans\_code", StringType, true),

StructField("item\_seq", IntegerType, true),

StructField("product\_cd", StringType, true),

StructField("amt", DoubleType, true),

StructField("disc\_amt", DoubleType, true),

StructField("add\_flg", IntegerType, true),

StructField("store", StringType, true),

StructField("emp\_num", IntegerType, true),

StructField("lane", IntegerType, true),

StructField("t\_stmp", StringType, true) ) )

val llSchema = StructType(Seq(StructField("seq\_num", IntegerType, true),

StructField("trans\_code", StringType, true),

StructField("loyalty\_cd", StringType, true),

StructField("store", StringType, true),

StructField("emp\_num", IntegerType, true),

StructField("lane", IntegerType, true),

StructField("t\_stmp", StringType, true) ) )

val ppSchema = StructType(Seq(StructField("seq\_num", IntegerType, true),

StructField("trans\_code", StringType, true),

StructField("promo\_cd", StringType, true),

StructField("store", StringType, true),

StructField("emp\_num", IntegerType, true),

StructField("lane", IntegerType, true),

StructField("t\_stmp", StringType, true) ) )

val ttDf = sqlContext.createDataFrame(TT, ttSchema)

val llDf = sqlContext.createDataFrame(LL, llSchema)

val ppDf = sqlContext.createDataFrame(PP, ppSchema)

ttDf.registerTempTable("ttStageTable")

llDf.registerTempTable("llStageTable")

ppDf.registerTempTable("ppStageTable")

sqlContext.sql("select \* from ttStageTable").collect

sqlContext.sql("select \* from llStageTable").collect

sqlContext.sql("select \* from ppStageTable").collect

sqlContext.sql("select tt.\*, s.store\_id from ttStageTable tt join mydb.store s on tt.store = s.store\_num").collect

def getTransId(ts:String, d\_store\_id:Int, d\_lane:Int, d\_trans\_seq:Int): Long = {

val daydate = ts.replaceAll("-","").split(" ")(0)

val store\_id = "%05d".format(d\_store\_id)

val lane = "%02d".format(d\_lane)

val trans\_seq = "%04d".format(d\_trans\_seq)

(daydate + store\_id.toString + lane.toString + trans\_seq.toString).toLong

}

val transId = getTransId(\_,\_,\_,\_)

val tid = sqlContext.udf.register("getTransId", transId)

a.select(tid(a("t\_stmp"), a("store\_id"), a("lane"), a("seq\_num")))

#ref:https://stackoverflow.com/questions/8131291/how-to-convert-an-int-to-a-string-of-a-given-length-with-leading-zeros-to-align