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),
```

```scala
        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
```

```scala
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
```