hive -e "drop table table\_name"

sqoop import --connect 'jdbc:sqlserver://DB\_Address:1433;databasename=DB\_name' --username testuser --password 1234 –table table\_name --hive-table pa\_soc.r2d2\_cv --create-hive-table --hive-import --m 1 --driver com.microsoft.sqlserver.jdbc.SQLServerDriver

hive -e "select p.FileName from  pa\_soc.r2d2\_cv p  left outer join pa\_soc.people\_content b on p.User\_GGID=b.uid  where date\_format(b.ts\_refresh,'yyyyMMdd')>date\_format(p.LastUpdate,'yyyyMMdd')"| sed 's/[\t]/,/g'  >  /exec/data/out/people.csv

cat people.csv | grep ".pdf" > people1.csv

cat people.csv | grep -v ".pdf" > people2.csv

awk '$0="/exec/data/out/pdf/PEOPLE\_DOC/"$0' /exec/data/out/people1.csv > /exec/data/out/people\_content

awk '$0="/exec/data/out/pdf/PEOPLE\_DOC/"$0".pdf"' /exec/data/out/people2.csv >> /exec/data/out/people\_content

rm -r /exec/data/out/r2d2\_cv

mkdir /exec/data/out/r2d2\_cv

hadoop fs -get `cat /exec/data/out/people\_content` /exec/data/out/r2d2\_cv

$spark-shell --packages com.databricks:spark-csv\_2.11:1.2.0 -i file.scala

$ spark-shell --packages com.databricks:spark-csv\_2.11:1.2.0

:cp /var/lib/spark/sqljdbc4-2.0.jar

val sqlContext = new org.apache.spark.sql.SQLContext(sc)

val df\_r2d2\_table = sqlContext.read.format("jdbc").option("url", “jdbc:sqlserver://DB\_Address:1433;databasename=DB\_name;user= testuser;password=1234;").option("driver", "com.microsoft.sqlserver.jdbc.SQLServerDriver").option("dbtable", "dbo.R2D2\_CV").option("user", " testuser ").option("password", "1234").load()

val hiveContext = new org.apache.spark.sql.hive.HiveContext(sc)

val df\_hive\_table = hiveContext.sql("select \* from pa\_soc.people\_content")

val df\_left\_join = df\_r2d2\_table.join(df\_hive\_table, df\_r2d2\_table("User\_GGID") === df\_hive\_table("uid"), "left\_outer")

val df\_left\_join\_filter = df\_left\_join.where(date\_format(col("LastUpdate"), "YYYYMMdd") < date\_format(to\_date(col("ts\_refresh")), "YYYYMMdd"))

val result = df\_left\_join\_filter.select("FileName")

sqlContext.setConf("spark.sql.tungsten.enabled", "false")

result.coalesce(1).write.format("com.databricks.spark.csv").save("/home/adpaapp/csv5")

result.write.save("/home/adpaapp/csv")                              --> saves as parquet file

result.rdd.repartition(1).saveAsTextFile("/home/adpaapp/text1")                             --> saves as textfile

This message contains information that may be privileged or confidential and is the property of the Capgemini Group. It is intended only for the person to whom it is addressed. If you are not the intended recipient, you are not authorized to read, print, retain, copy, disseminate, distribute, or use this message or any part thereof. If you receive this message in error, please notify the sender immediately and delete all copies of this message.