Task 3

Spark ETL: Option 3 with RDD, Dataframe and Spark SQL
Notebook:
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/1092176685531650/3530701261005479/6776489139542437/latest.html

- Read file

```
txF = sc.textFile(“<hdfs dir>/transactions.csv”)
balF = sc.textFile(“<hdfs dir>/balance.csv”)
```

- Generate key value from a flat string

```
from pyspark.sql import Row
tx1 = txF.map(lambda x: Row(account_id=x.split(",")[0], amt=x.split(",")[1])).toDF();
tx1.registerTempTable("tranx")
bal1 = balF.map(lambda x: Row(account_id=x.split(",")[0],
balance=int(x.split(",")[1]))).toDF()
bal1.registerTempTable("accBal")
```

- Aggregate transaction amount for all the transactions of individual accounts

```
tx2 = sqlContext.sql("select account_id, cast(sum(amt) as int) as bal from tranx group by
account_id")
tx2.registerTempTable("aggTranxBal")
```

- Join balance and aggregated transactions RDDs

```
joinedDf = sqlContext.sql("select b.account_id, t.bal, b.balance from aggTranxbal t join
accBal b on t.account_id = b.account_id" )
joinedDf.registerTempTable("joinedDf")
```

- Filter all the accounts for which reconciliation doesn’t match with current balance

```
errorAccounts = sqlContext.sql("select * from joinedDf j where j.bal != j.balance" )
```

- Save the errorAccounts RDD in HDFS

```
errorAccounts.map(lambda x: str(x[0]) + "," + str(x[1]) + "," +
str(x[2])).saveAsTextFile("<HDFS path>")
```