**Task 1**

**Objective:** Recall basic commands to carry out common operations

1. Copy data files from local system to HDFS
2. Carry out following operations on Spark
   1. Read a csv file
   2. Transform a line of flat string into meaningful fields
   3. Aggregate
   4. Join
   5. Filter
   6. Save data back to filesystem
3. Display content of an HDFS file

**Solution approach**

Copy files transactions.csv and balance.csv to hdfs

hadoop fs -copyFromLocal <local dir>/transactions.csv <hdfs dir>/

hadoop fs -copyFromLocal <local dir>/balance.csv <hdfs dir>/

Display content of an HDFS file

hadoop fs -cat <hdfs file path>

**Spark ETL: Option 1 with only RDD**

**Notebook:** <https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/1092176685531650/3530701261005462/6776489139542437/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

tx1=txF.map(lambda x: (x.split(",")[0], int(x.split(",")[1])))

bal1 = balF.map(lambda x: (x.split(",")[0], x.split(",")[1]))

* Aggregate transaction amount for all the transactions of individual accounts

tx2 = tx1.reduceByKey(lambda x,y: x+y)

* Join balance and aggregated transactions RDDs

joinedRdd = bal1.join(tx2)

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

errorAccounts = joinedRdd.filter(lambda x: int(x[1][0]) != int(x[1][1]) )

* Save the errorAccounts RDD in HDFS

errorAccounts.saveAsTextFile("<HDFS path>")