**Task 2**

**Spark ETL: Option 2 with RDD and Dataframe**

**Notebook:** <https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/1092176685531650/3530701261005471/6776489139542437/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();

bal1 = balF.map(lambda x: Row(account\_id=x.split(",")[0], balance = int(x.split(",")[1]))) .toDF()

* Aggregate transaction amount for all the transactions of individual accounts

tx2 = tx1.groupBy("account\_id").agg(sum("amt").alias("bal"))

**How to aggregate with an alias to agg column:**

<https://stackoverflow.com/questions/33882894/sparksql-apply-aggregate-functions-to-a-list-of-column>

<https://stackoverflow.com/questions/36719039/sum-operation-on-pyspark-dataframe-giving-typeerror-when-type-is-fine>

from pyspark.sql.functions import sum as \_sum

tx2 = tx1.groupBy("account\_id").agg(\_sum("amt").alias("bal"))

* Join balance and aggregated transactions RDDs

joinedDf = tx2.join(bal1, tx2.account\_id == bal1.account\_id)

**When you have multiple columns with same name, mention just name**

**i.e.** joinedDf = tx2.join(bal1, ‘account\_id’)

<https://docs.databricks.com/spark/latest/faq/join-two-dataframes-duplicated-column.html>

**Join with multiple columns**

**i.e.** joinedDf = tx2.join(bal1, (tx2.account\_id ==bal1.account\_id) & (tx2.bal==bal1.balance))

<https://stackoverflow.com/questions/33745964/how-to-join-on-multiple-columns-in-pyspark>

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

errorAccounts = joinedDf.filter(joinedDf.bal != joinedDf.balance)

* Save the errorAccounts RDD in HDFS

errorAccounts.rdd.saveAsTextFile("<HDFS path>")

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

**Default is parquet**: errorAccounts.save("<HDFS path>")

**Multiple json files**: errorAccounts.write.format("json").save("file:///home/hduser/df2")

**Single file:** errorAccounts.repartition(1).write.format("json").save( "file:///home/hduser/df3")