Below is an example code that converts last names (2nd column) to upper case in a data table that is separated by a question mark.

Given data in a file inputNames.txt, located in /home/ec2-user/inputNames.txt

Alexander?Skarsgard

Ali?Larter

Alicia?Keys

Amanda?Bynes

Amanda?Seyfried

America?Ferrera

Amy?Adams

Python code placed in /home/ec2-user/upperLName.py (**note** that split and joined on '\t', not on '?', even though the original data is separated by ?)

#!/usr/bin/python

import sys

for line in sys.stdin:

line = line.strip().split('\t')

FName= line[0]

LName = line[1]

LName = LName.upper()

print '\t'.join([FName, LName])

(note the ? as the delimiter here)

**CREATE TABLE UserName( FName Varchar(25), LName Varchar(25))**

**ROW FORMAT DELIMITED FIELDS TERMINATED BY '?' STORED AS TEXTFILE;**

(load the data)

**LOAD DATA LOCAL INPATH '/home/ec2-user/inputNames.txt' OVERWRITE INTO TABLE UserName;**

**ADD FILE /home/ec2-user/upperLName.py;**

(create a transformed output)

**SELECT TRANSFORM (FName, LName) USING 'python upperLName.py' AS (FName,LName) FROM UserName;**

(make another table)

**CREATE TABLE UserNameLU( FName Varchar(25), LName Varchar(25))**

**ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE;**

(Load the same upper-last-name data into the new table, this performs same transformation again)

**INSERT OVERWRITE TABLE UserNameLU SELECT TRANSFORM (FName, LName) USING 'python upperLName.py' AS (FName,LName) FROM UserName;**

Example that describes ORDER BY MapReduce (choose earliest year and average grade by first name of a student)

SELECT MIN(Year), AVG(Grade)

FROM Student

GROUP BY FirstName

ORDER BY AVG(Grade)

Mapper1: (key: FirstName, value: Year, Grade).

For an input block of data, for every student record that your code identifies, set the FirstName as the key and set Year, Grade as a value.

Reducer1: For each FirstName received, compute and output the minimum of all Year values and the average of all Grade values. Optionally, you could skip outputting the key here and produce just the values.

Second pass (required for sorting) – applied to the output of previous pass.

Mapper2: (key: AvgGrade, value: MinYear)

For an input block of data, for each record with minimum year and average grade information, set the average grade as the key and the corresponding minimum year as the value.

Modify the partitioner to a custom range function in order to enable key-based sorting [you must use that for sorting]

Reducer2: For each AvgGrade received, output the MinYear values as a list (e.g., 3.73 2012 2014 2011). Note that the result is not going to be exactly as the SQL query output because of duplicate AvgGrade entries.