User Manual

For

Big Data Integration and Analysis

llama
BIG DATA INTEGRATION
& ANALYSIS

Presented by

Gaurav Kumar

Héctor Ugarte

Miguel Mármol

Tina Boroukhian

University of Bonn

Summer 2015

Table of contents

---

1. Introduction

These days, data streams from each and every activity of daily life: from phones, credit cards, televisions and computers; from sensor-equipped buildings, GPS, trains, buses, planes, bridges, factories, and so on. The data flows so fast that the total accumulation of the past two years—a zettabyte—dwarfs the prior record of human civilization. This huge amount of data is very important as it contains a lot of useful information and considering the volume, velocity and variety of data, cleaning and analyzing big data is a big challenge.

A real example of such a challenge can be seen in e-commerce companies, such as Amazon, where they have huge amounts of customer-related data. This data is crucial to any company for which reason they are ready and eager to spend a big portion of their budget in analyzing data so they can, among others, establish solid predictive models. For the example of e-commerce companies, one concern is to list the most sought products by their customers so they can predict which class of customers (per age for example) tends to by what in which period of time. [1][2]

1.1 Why Integration & Analyzing Data?

Extracting, cleaning, and loading data are three core steps in the Data Integration process. Data reshaping programs are difficult to write because of their complexity, but they are required because each analytic tool expects data in a very specific form and to get the data into that form typically requires a whole series of cleaning, normalization, reformatting, integration, and restructuring operations.

Analyzing Big Data is used to find meaning and discover hidden relationships in Big Data. The technological advances in storage, processing, and analysis of Big Data include the rapidly decreasing cost of storage and CPU power in recent years; the flexibility and cost-effectiveness of datacenters and cloud computing for elastic computation and storage; and the development of new frameworks such as Hadoop and Spark, allowed users to take advantage of these distributed computing systems storing large quantities of data through flexible parallel processing. [1],[3],[4].

1.2 Apache Hadoop

Apache Hadoop [5] is an open-source software framework written in Java for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware.

The core of Apache Hadoop consists of:

● Storage part: Hadoop Distributed File System (HDFS).

● Processing part: Hadoop MapReduce.

Hadoop splits files into large blocks and distributes them amongst the nodes in the cluster. To process the data, Hadoop MapReduce transfers packaged code for nodes to process in parallel, based on the data each node needs to process. Hadoop (MapReduce) alongside the built-in solutions like Hive and Pig, have been widely used for years now intensively in batch processing chains. One such a chain are ETL(-like) data integration programs.

1.3 Apache Spark

Apache Spark [6] is a cluster computing platform designed to be fast and general purpose. On the speed side, Spark extends the popular MapReduce model to efficiently support more types of computations, including interactive queries and stream processing. Speed is for instance very important in interactive computations where responses are ultimately needed in scale of few seconds. This includes querying dataset and running iterative programs, such as the ones found in Machine Learning. Therefore, Spark came with native in-memory data processing that speeds-up hugely these types of computing.

1.4 Authorized use permission

This system is designed for lab project study in Enterprise Information Systems.

2. System Summary

2.1 System Configuration

- Ubuntu 14.4
- Hadoop 2.6
- Spark 1.4
- Scala 2.1
- Maven
- Eclipse IDE
- Java JDK 1.8

2.2. Architecture Diagram

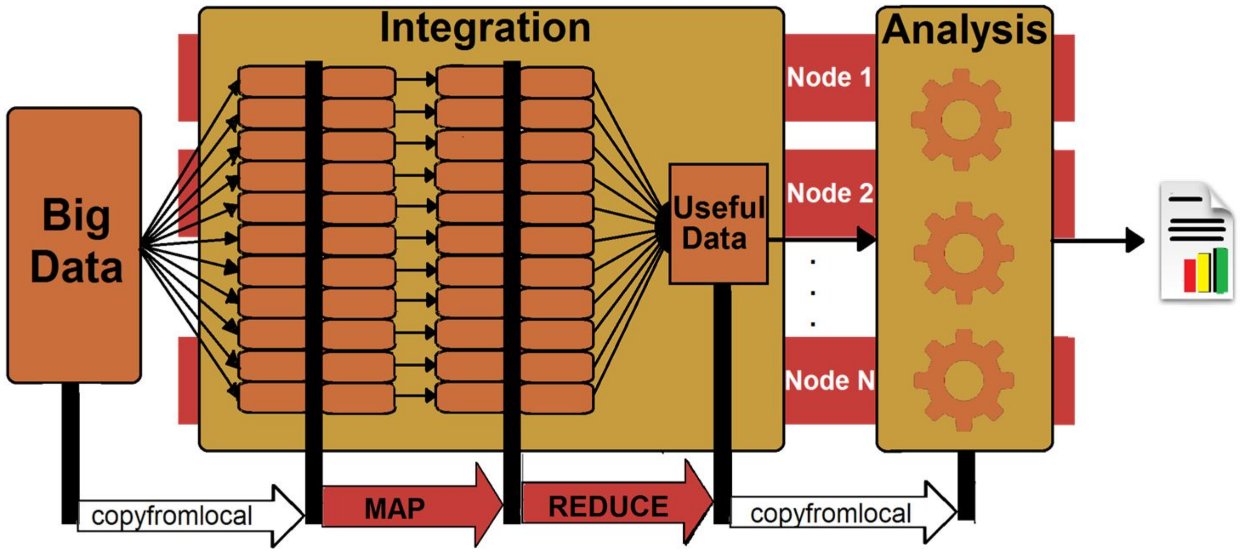


Figure 1: Architecture of Integration and Analysing of Data [7].

----

3. INTEGRATION JOB SPECIFICATION LANGUAGE

We propose a language to users are able to describe the integration job.

We name it *Integration Job Specification Language***,** *IJSL* for short.

3.1. IJSL Grammar

IJSL script consists of keywords and parameters:

KEYWORDS

There are 10 keywords (which can be written in uppercase or lowercase). For clarity, will write them on uppercase throughout our report:

INPUTFILE, OUTPUTFILE, SEPARATOR, PROJECTEDCOLUMNS, PROJECTEDCOLUMNSNAMES, MERGE, SPLIT, CASE, FORMATDATES, RESTRICTION, EMPTY

OBLIGATORY	**OPTIONAL**
INPUTFILE	MERGE
OUTPUTFILE	SPLIT
SEPARATOR	CASE
PROJECTEDCOLUMNS	FORMATDATES
PROJECTEDNAMES	RESTRICTION

Table 2: Some keywords are obligatory and others optional

PARAMETERS

Parameters are defined with the following structure:

Par1|Par2|…

If we don't want to define parameters **on optional keywords,** we use the keyword:

```
EMPTY
```

Example: **MERGE** EMPTY

or just don't write that statement.

INPUTFILE

Defines the input CSV file name.

Example: **INPUTFILE** Customers.csv

OUTPUTFILE

Defines the output CSV file name.

Example: **OUTPUTFILE** CustomersOutput.csv

SEPARATOR

Defines the delimiter of the input file. It can only one character.

Most common used ones are Comma (,) Semi-colon (;) Pipe (|) and Caret (^)

Example: **SEPARATOR** ,

PROJECTEDCOLUMNS

Defines the index of the columns to be projected (with 0 being the first index).

Part1: First projected column.

Part2: Second projected column.

…

PartN: N projected column.

Example: **PROJECTEDCOLUMNS** 1|3

PROJECTEDNAMES

Defines the names of the columns to be projected.

Part1: First projected column name.

Part2: Second projected column name.

…

PartN: N projected column name.

Example: **PROJECTEDNAMES** Name|City

[MERGE]

Defines the index of the two columns to be merged and the merge character.

Part1: First column index.

Part2: Second column index.

Part3: Merge character.

Example: **MERGE** 0|1|

[SPLIT]

Defines the index of column to be splitted and the character.

Part1: Column index.

Part2: Split character.

Example: **SPLIT** 2|_

[CASE]

Defines the index of column to upper or lower the case

Part1: Column index.

Part2: 0 for UPERCASE, 1 for LOWERCASE

Example: **CASE** 1|0

[FORMAT]

Defines the index of date column to be formatted.

Part1: Column index.

Part2: DD/MM/YYYY MM/DD/YYYY YYYY/MM/DD

Example: **FORMAT** 3|MM/DD/YYYY

[RESTRICTION]

Defines the index of column to be restricted (filtered), the operator used and the value.

Part1: Column index.

Part2: For numeral values: =, <>, >,<,>=,<= For textual values: EQUAL, NOT EQUAL, CONTAINS

Part3: value

Example: **RESTRICTION** 0|>=|20|0|<|50

3.2. EXAMPLES:

Our example CSV input file is called Customers.csv, and has the following header, 6 columns: Customer_ID|Name|Address|City|ZipCode|Phone

A. We want as output 3 columns: Customer_ID, Address, City where Customer_ID values are greater than or equal to 20 and less than 50.

```
INPUTFILE Customers.csv
OUTPUTFILE Output.csv
SEPARATOR |
PROJECTEDCOLUMNS 0|2|3
PROJECTEDNAMES Customer_ID|Address|City
RESTRICTION 0|>=|20|0|<|50
```

B. We want as output all the columns, but rename some of them: Customer_ID -> ID, ZipCode -> Postal_Code. And change Name case to capital letters.

```
INPUTFILE Customers.csv
OUTPUTFILE Output.csv
SEPARATOR |
PROJECTEDCOLUMNS 0|1|2|3|4|5
PROJECTEDNAMES ID|Name|Address|City|Postal_Code|Phone
MERGE EMPTY
SPLIT EMPTY
CASE 1|0
FORMATDATES EMPTY
RESTRICTION EMPTY
```

C. We want as output only Column Name, but we want to split it on two columns First_Name and Last_Name.

```
INPUTFILE Customers1.csv
OUTPUTFILE Output.csv
SEPARATOR |
PROJECTEDCOLUMNS 1|1000
PROJECTEDNAMES First_Name|Last_Name
MERGE EMPTY
SPLIT 1|
CASE empty
FORMATDATES EMPTY
RESTRICTION EMPTY
```

---

4. Application

In order to install the application we need to unzip the file llama.zip into the path /home/hduser/Desktop/

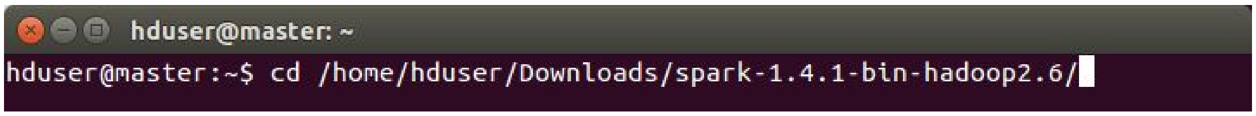
It will copy the following files:

- llama.jar
- llama.jpg
- Settings

4.1. Running it

On linux terminal:

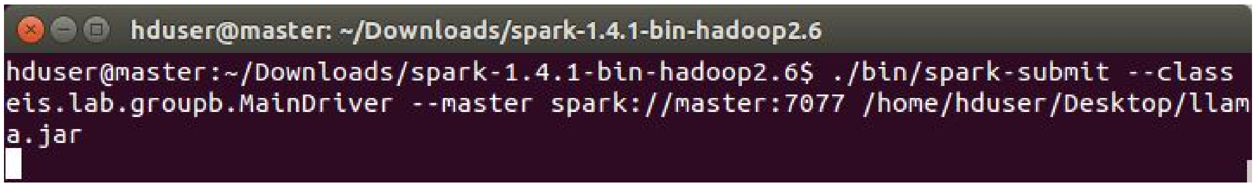
Locate the path where spark is installed, in this case /home/miguel/Downloads/spark-1.4.1-bin-hadoop2.6/



```
hduser@master: ~
hduser@master:~$ cd /home/hduser/Downloads/spark-1.4.1-bin-hadoop2.6/
```

and execute the following command

./bin/spark-submit --class eis.lab.groupb.MainDriver --master spark://master:7077 /home/hduser/Desktop/llama.jar



```
hduser@master: ~/Downloads/spark-1.4.1-bin-hadoop2.6
hduser@master:~/Downloads/spark-1.4.1-bin-hadoop2.6$ ./bin/spark-submit --class
eis.lab.groupb.MainDriver --master spark://master:7077 /home/hduser/Desktop/llam
a.jar
```

Options:

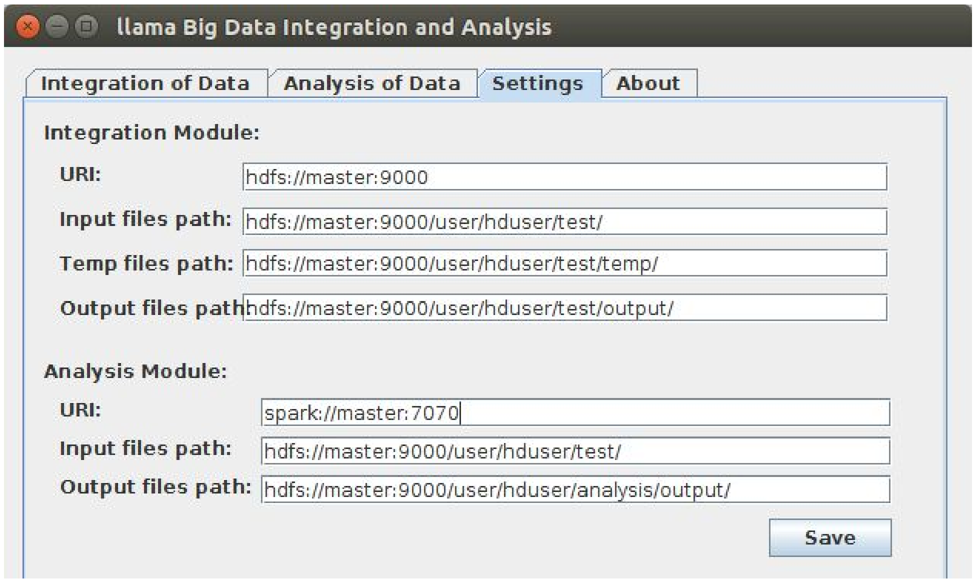
--class: Class that contains the main function.

--master: Location of the spark cluster. It should have the option *local* if we want to use spark locally.

Finally we have to indicate the path where the jar is located.

4.2 Settings

The application has the following list of settings, as shown in Figure 2, to edit them we have to go to the *Settings* tab and the following form will appear.



Integration Module:

URI:	hdfs://master:9000
Input files path:	hdfs://master:9000/user/hduser/test/
Temp files path:	hdfs://master:9000/user/hduser/test/temp/
Output files path	hdfs://master:9000/user/hduser/test/output/

Analysis Module:

URI:	spark://master:7070
Input files path:	hdfs://master:9000/user/hduser/test/
Output files path:	hdfs://master:9000/user/hduser/analysis/output/

Figure 2

Integration Module	
URI	URI where the HDFS is
Input files path	Path for the input files
Temp files path	Path for the temp files
Output files path	Path for the output files

Analysis Module	
URI	URI where the HDFS is
Input files path	Path for the input files
Output files path	Path for the output files

4.3. Data Integration

In this section we will describe the Data Integration phase.

We have three possible ways to do an integration job:

- Specifying values manually.
- Loading an existing IJSL script.
- Writing a IJSL script.

4.3.1. Specifying values manually

In Figure 3:

1) Select the first tab named “Integration of Data”.

2) Select a file from the list of files already uploaded on HDFS.

3) Click “Select” button.

4) Define the separator character that we are using, by default the value is comma (,)

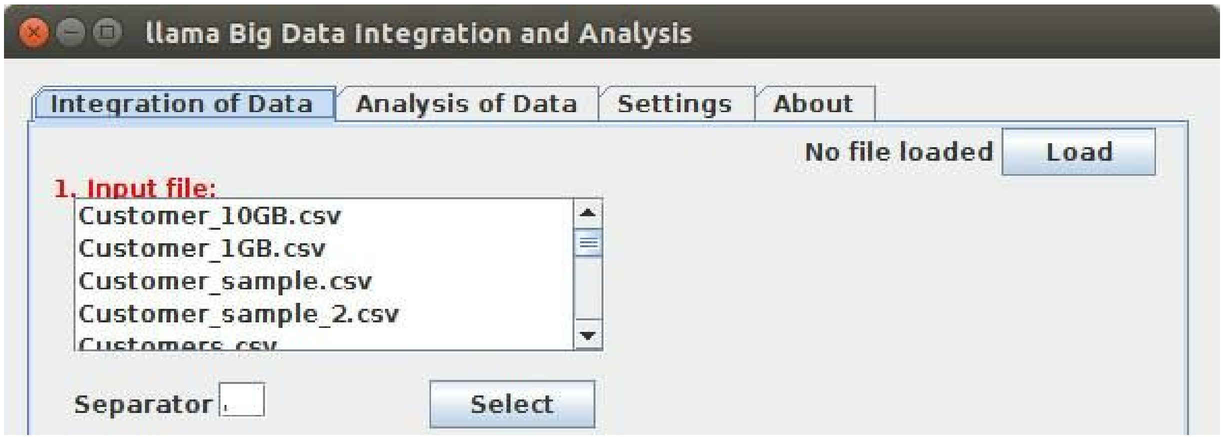


Figure 3

In Figure 4: a list of “Desired columns” will appear from which:

1) select the needed columns for the integration,

2) Click the “Select” button.

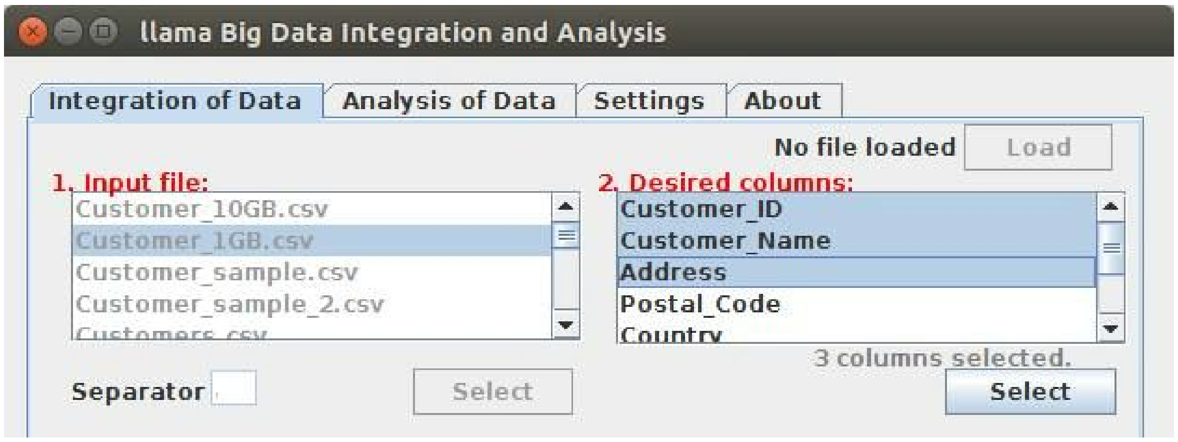


Figure 4

In Figure 5, as you can see three sections appears corresponding respectively to: Transformation Operations, Restriction Operations and Output files. 3 and 4 are optional 5 is obligatory.

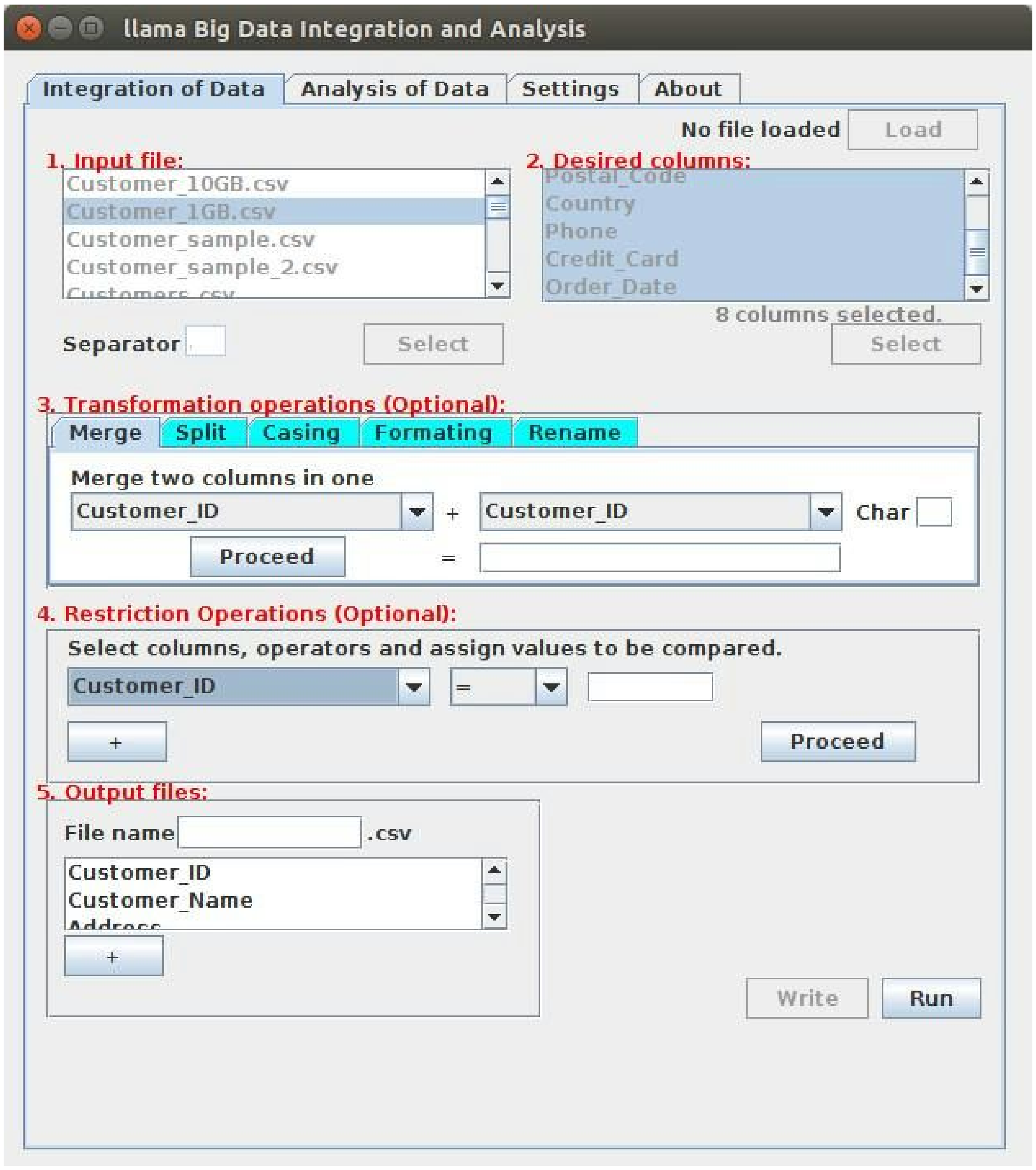


Figure 5

4.3.1.1. Transformations Operations:

Four transformations operations using Mapreduce are offered by the software: Merge columns, Split columns, Case of columns, and Formatting of date columns and one for Rename headers.

In Figure 6, For merging:

1) Select the Merge tab in Transformation Operations.

2) Select the two columns which you want to merge,

3) Define the merging character the used by default is blank space.

4) Write the name of the new column in the Textbox located after the equal sign '='.

5) Click the Proceed button.

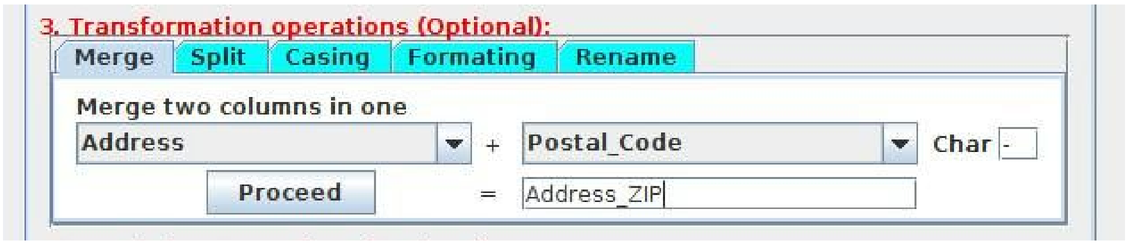


Figure 6

In Figure 7, For splitting:

1) Select the column Split tab

2) Select the column name to be splitted.

3) Write the name of the new two columns after the equal sign '='. ,

4) Define the splitting character the used by default is blank space.

5) Click "Proceed".

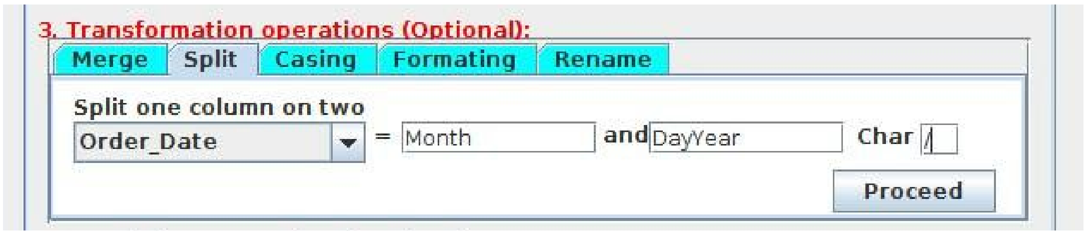


Figure 7

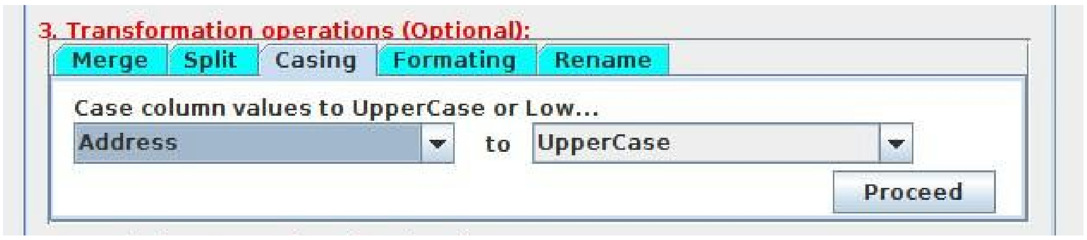
In Figure 8, for Casing:

1) Select casing tab

2) Select the desired column

3) Select UpperCase or LowerCase

4) Click "Proceed"



Case column values to UpperCase or Low...

Address to UpperCase

Figure 8

In Figure 9, for Formatting:

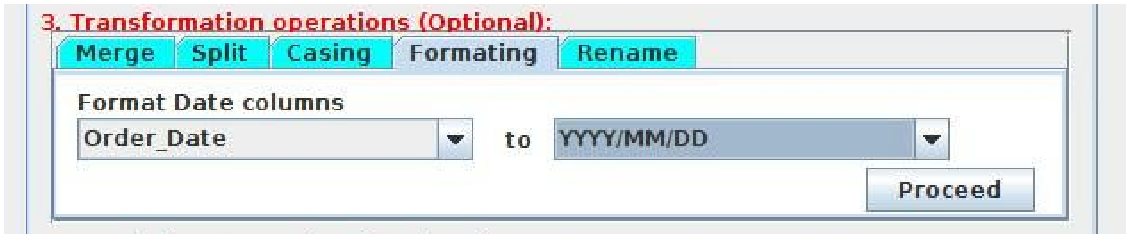
1) Select Formatting tab.

2) Select the desired column to format, it must be a date column

3) Select the desired date format for related column.

Three date formats are supported: DD/MM/YYYY MM/DD/YYYY YYYY/MM/DD

4) Click "Proceed"



Format Date columns

Order_Date to YYYY/MM/DD

Figure 9

In Figure 10, for Renaming columns:

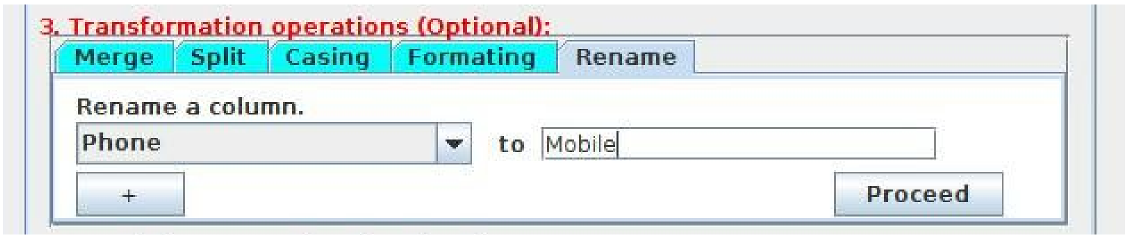
1) Select Rename tab

2) Select the desired column to be renamed.

3) Write the new desired name.

4) Click "Proceed"

It is also possible to rename more columns clicking on the "+" button.



Rename a column.

Phone to Mobile

Figure 10

4.3.1.2. Restriction operation

In Restriction Operations section, we can restrict the output according to some criterias.

For numeric values: =, <>, >,<,>=,<=

For textual values: EQUAL, NOT EQUAL, CONTAINS.

In Figure 11:

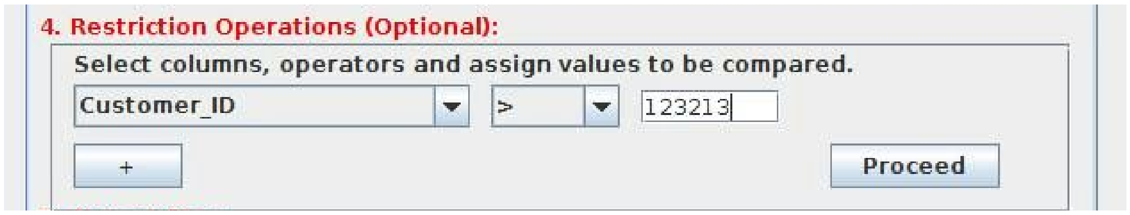
1) Select the desired column

2) Select the criteria to compare

3) Write the value which you want to compare it.

4) Click "Proceed" button.

It is also possible to add more restrictions clicking on the "+" button.



4. Restriction Operations (Optional):

Select columns, operators and assign values to be compared.

Customer_ID > 123213

Figure 11

4.3.1.3. Output files

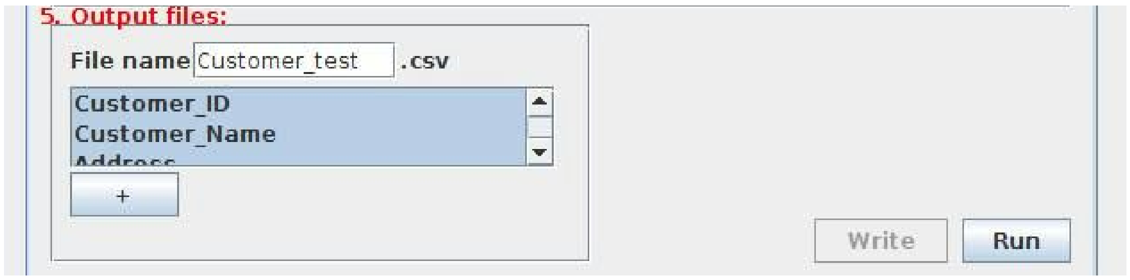
In Figure 12, we define the output files.

1) Select the desired columns

2) Write the desired name for output file.

3) Press the Run button.

It is also possible to generate more files clicking on the "+" button.



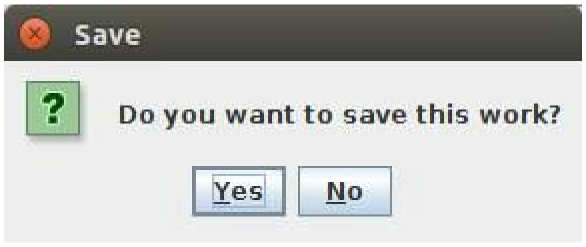
5. Output files:

File name Customer_test .csv

Customer_ID
Customer_Name

Figure 12

After the integration job is done one form is open, as shown in Figure 13, if you want to save it as a IJSL script click "Yes" otherwise click "No".



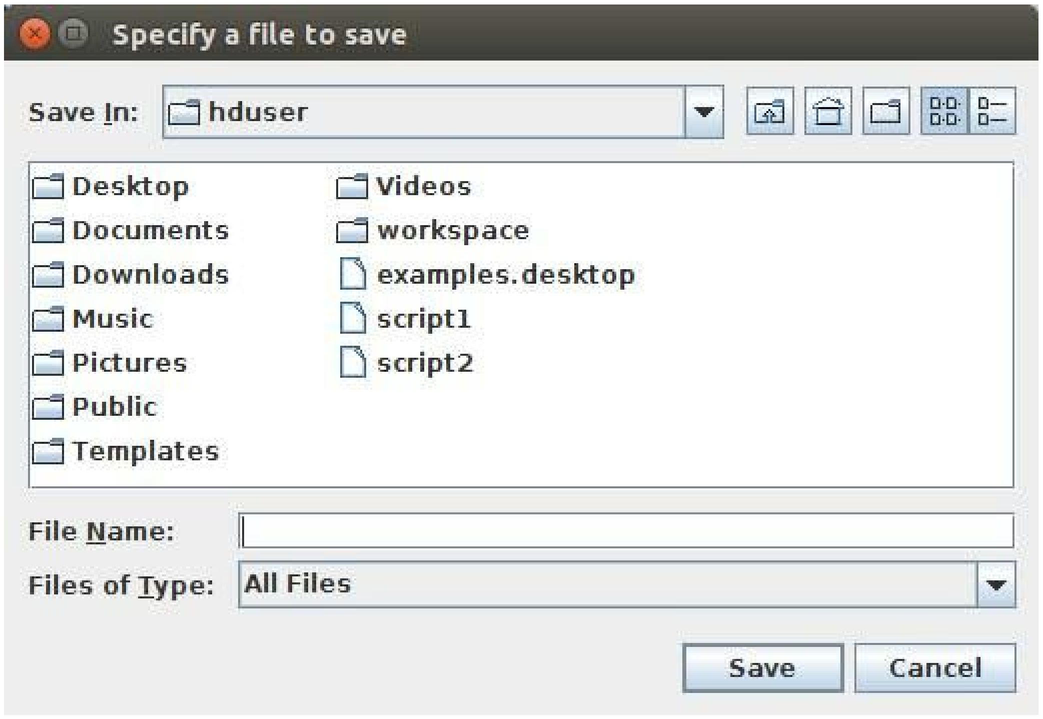
Do you want to save this work?

Figure 13

In Figure 14, It will appear one dialog:

1) Write the name of the file

2) Click "Save".



Save In: hduser

File Name:

Files of Type: All Files

Figure 14

4.3.2. Loading an existing IJSL script

Also you can Load an existing IJSL script file and Run it.

In Figure 15,16:

1) Click “Load” button,

2) Select file.

3) Click “Load” button.

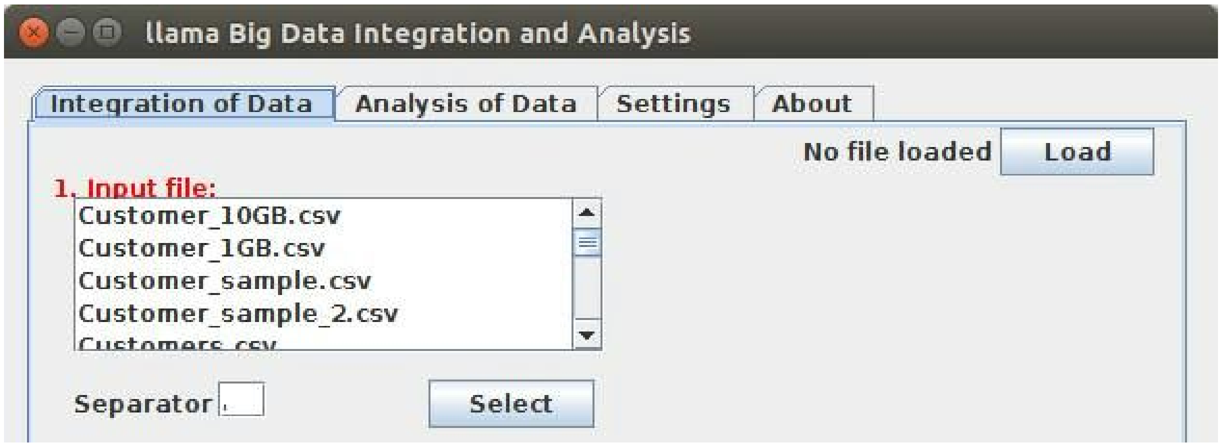


Figure 15

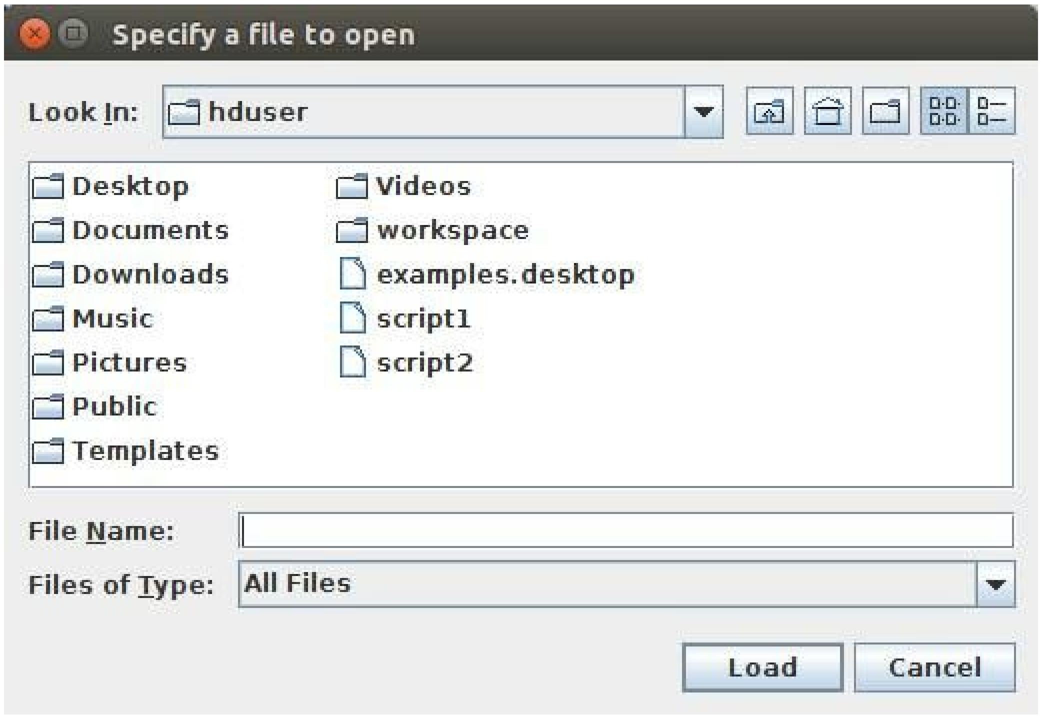
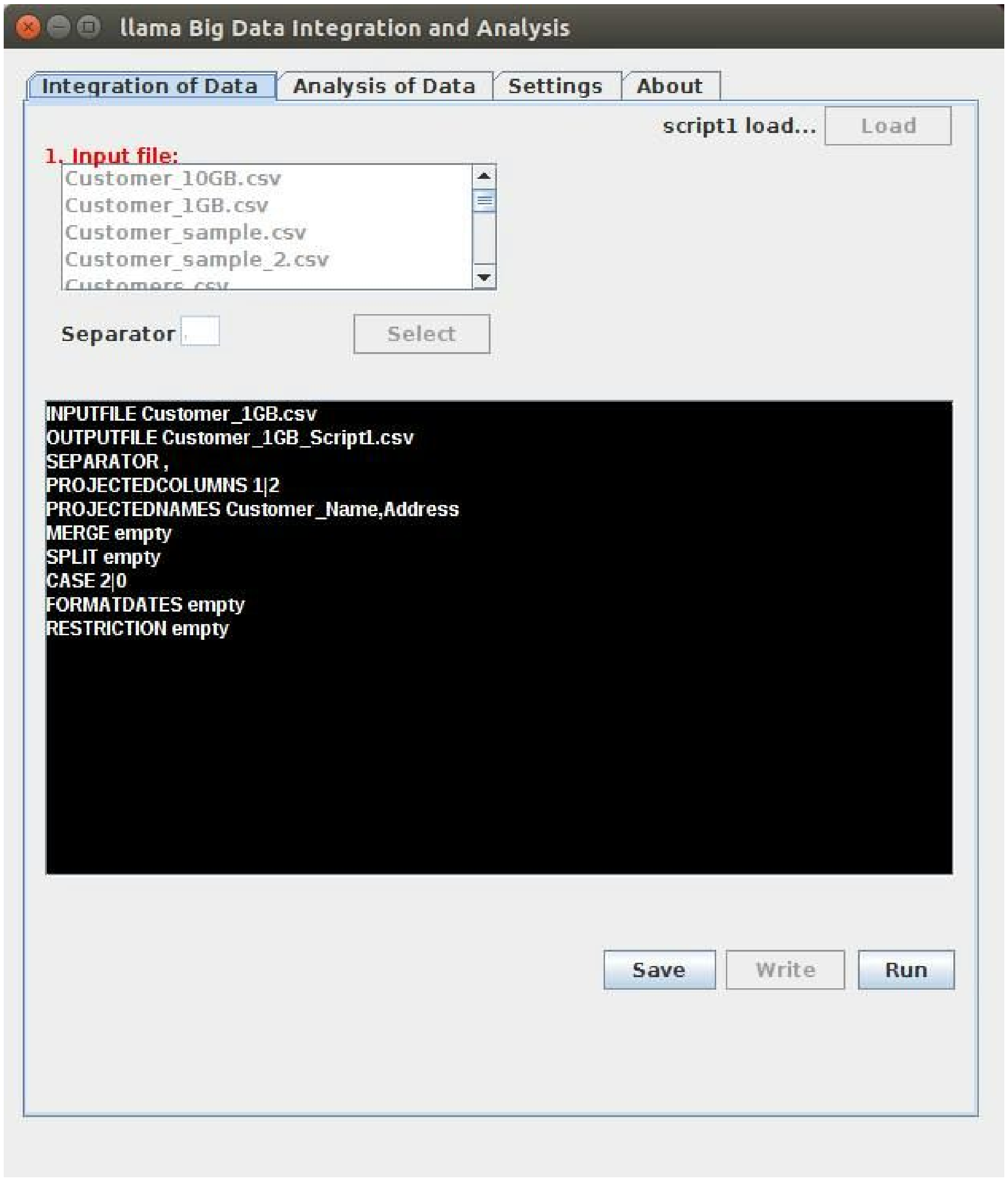


Figure 16

In Figure 17:

Verify the IJSL script. Modify it if needed.

1) Click run.



```
INPUTFILE Customer_1GB.csv
OUTPUTFILE Customer_1GB_Script1.csv
SEPARATOR ,
PROJECTEDCOLUMNS 1|2
PROJECTEDNAMES Customer_Name,Address
MERGE empty
SPLIT empty
CASE 2|0
FORMATDATES empty
RESTRICTION empty
```

Figure 17

4.3.3. Writing a IJSL script

The structure of the grammar for the IJSL script is explained at 4 .

In Figures 18,19:

1) Click “Write”

2) Write the IJSL script

3) Click “Run”

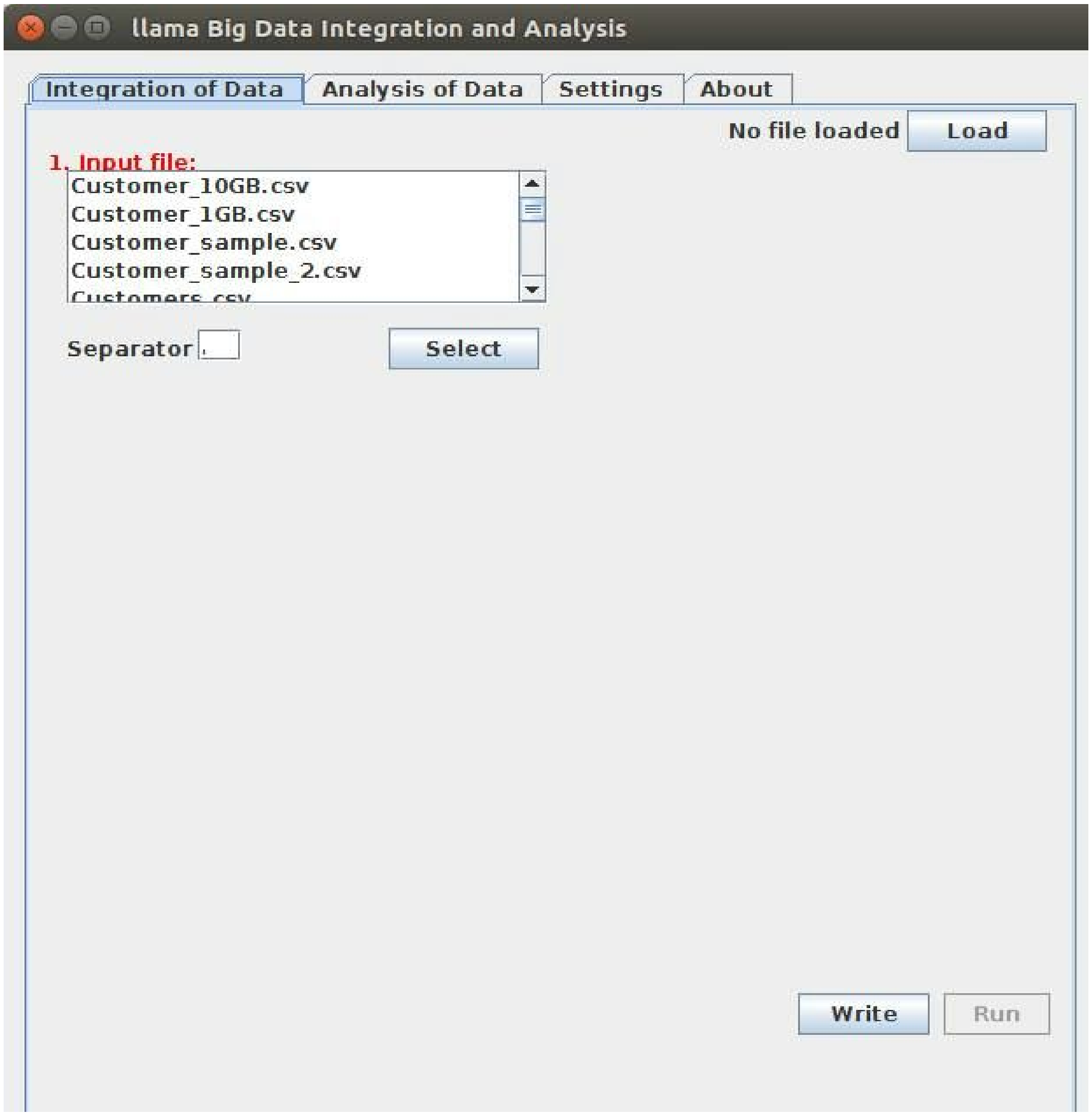
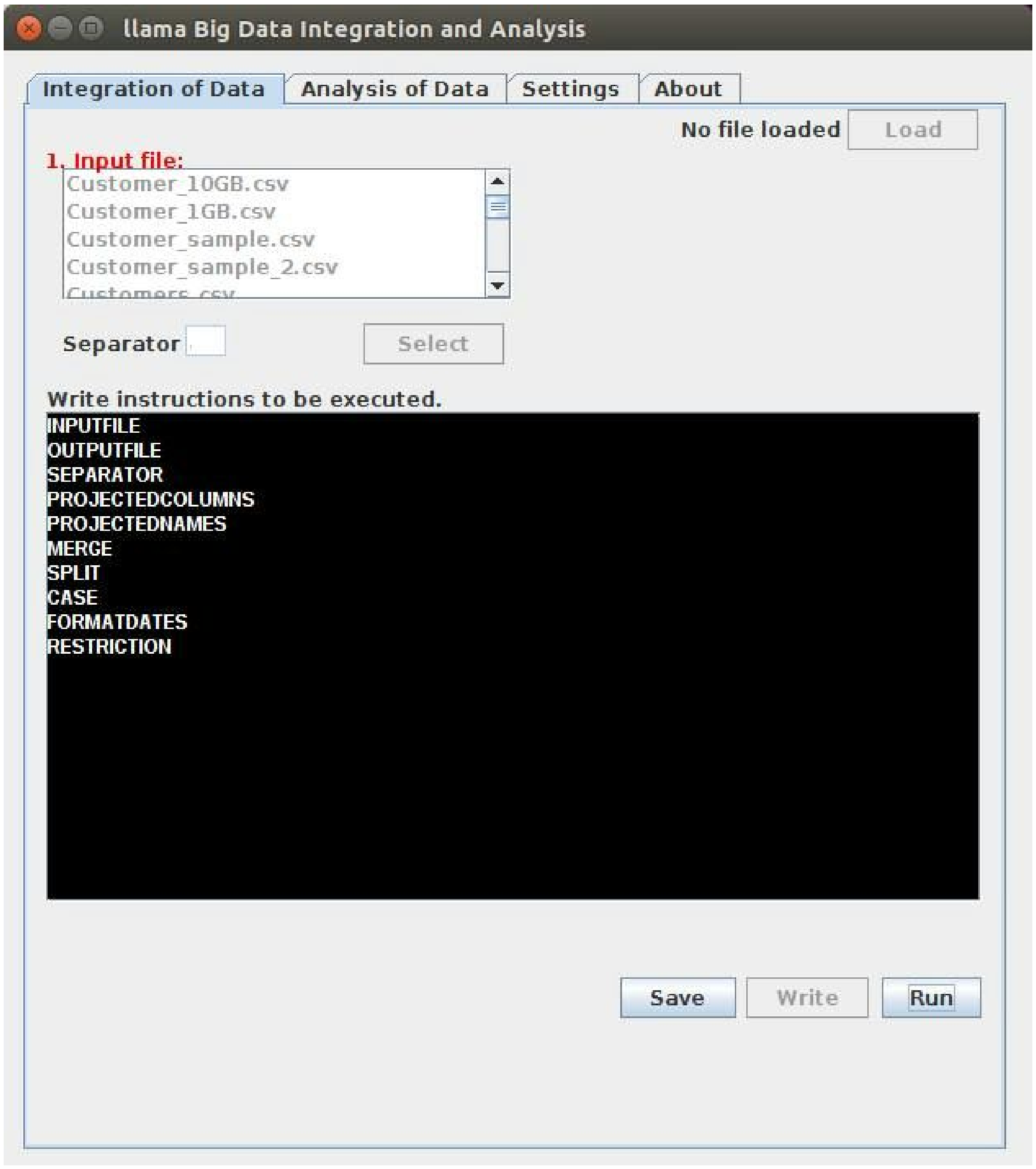


Figure 18



```
INPUTFILE
OUTPUTFILE
SEPARATOR
PROJECTEDCOLUMNS
PROJECTEDNAMES
MERGE
SPLIT
CASE
FORMATDATES
RESTRICTION
```

Figure 19

In Figure 20, also we can save our new IJSL script.

1) Click "Save".

2) Write the name of the file

3) Click "Save"

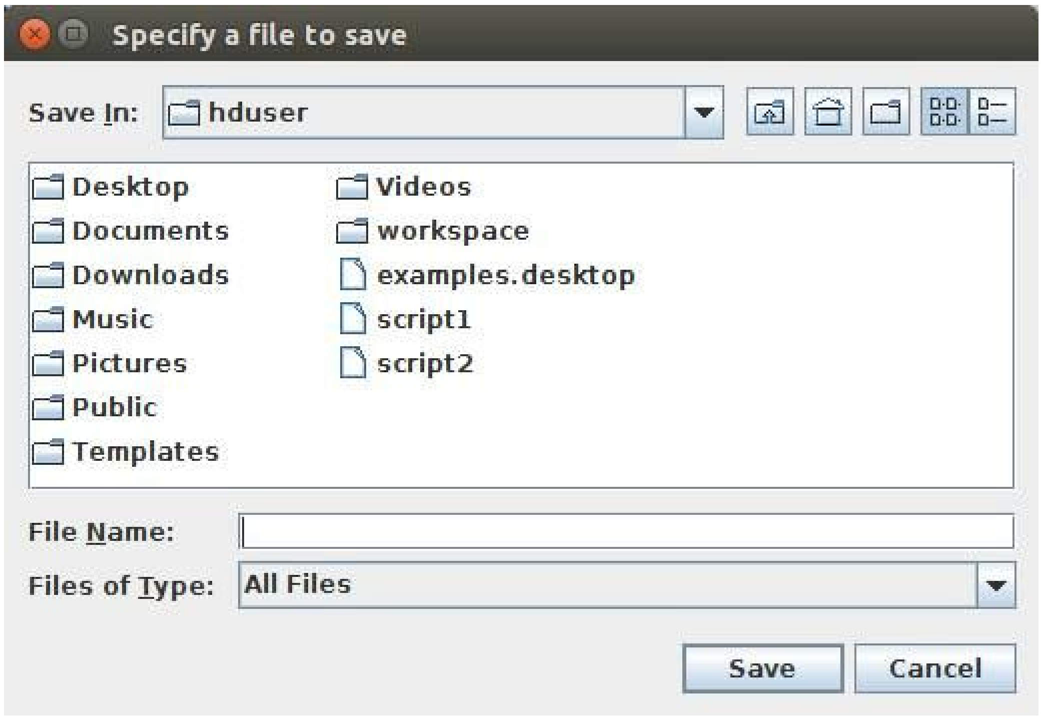


Figure 20

4.4. Data Analysis

In this section we will describe the Data Analysis phase.

4.4.1 Load files from Hadoop File System

The system will upload the list of files that are currently in the Input Directory that is specified in the Settings file. As shown in Figure 21.

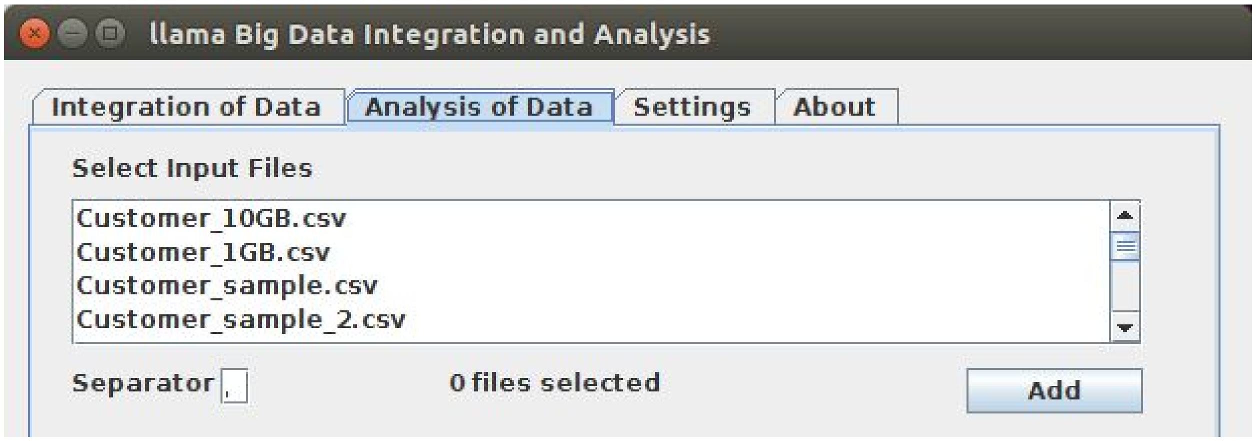


Figure 21

4.4.2 Selecting files from Hadoop File System

In Figure 22:

1) Define the separator (default is comma).
2) Select at least one file
3) Click button Add.

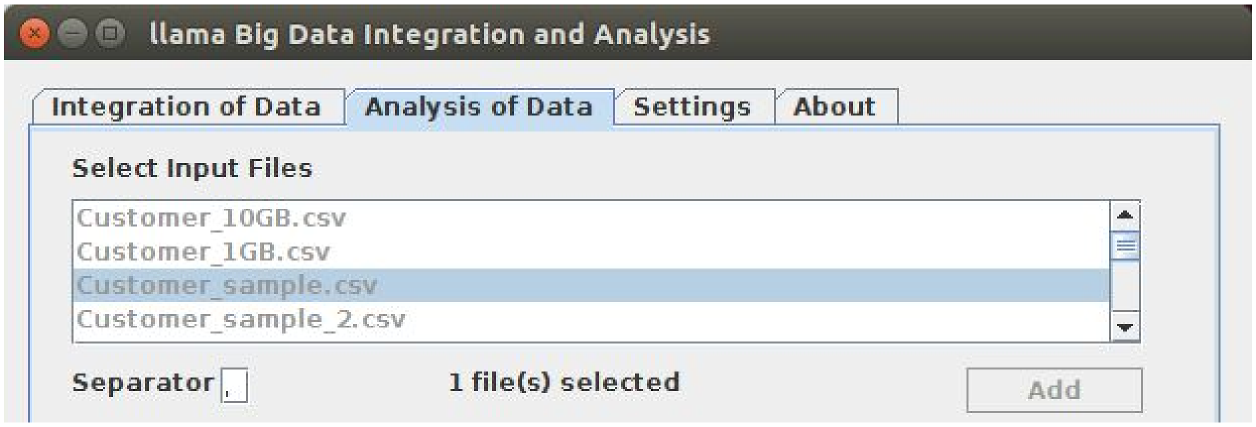


Figure 22

4.4.3 Displaying field names from selected files

The table will be registered in spark with the name of the file and it will be display among with its field names to help the user formulating the queries. As shown in Figure 23.

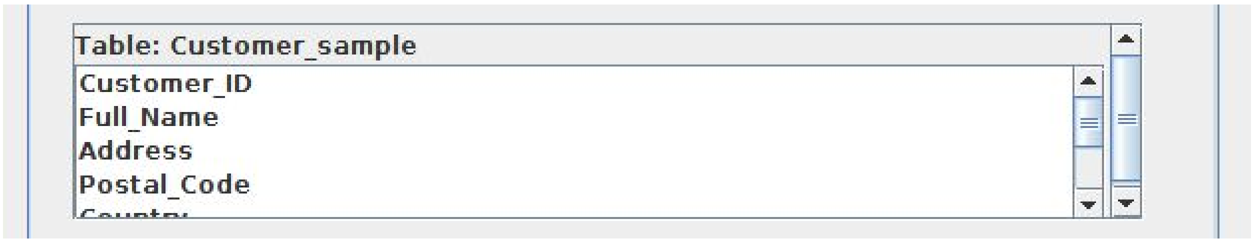


Figure 23

4.4.4 Input query

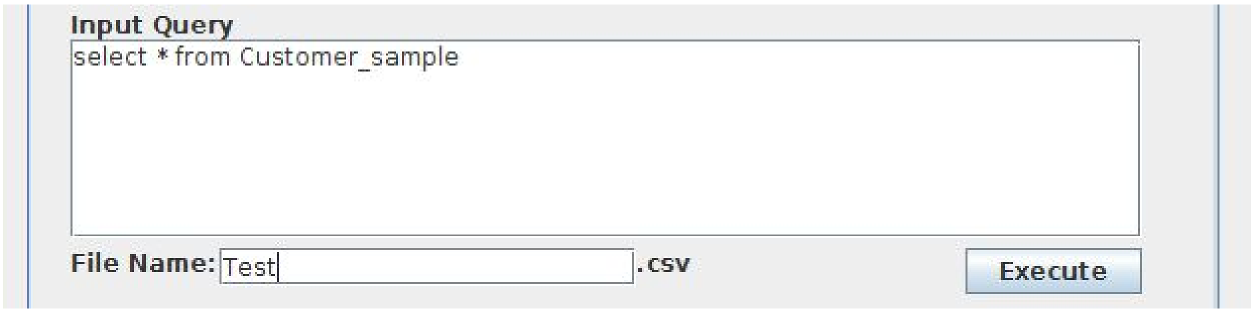
In Figure 24, a text area will be displayed for the user

---

1) Input the desired query.

2) Input a name to save the results in the HDFS.

2) Click "Execute" button.



Input Query

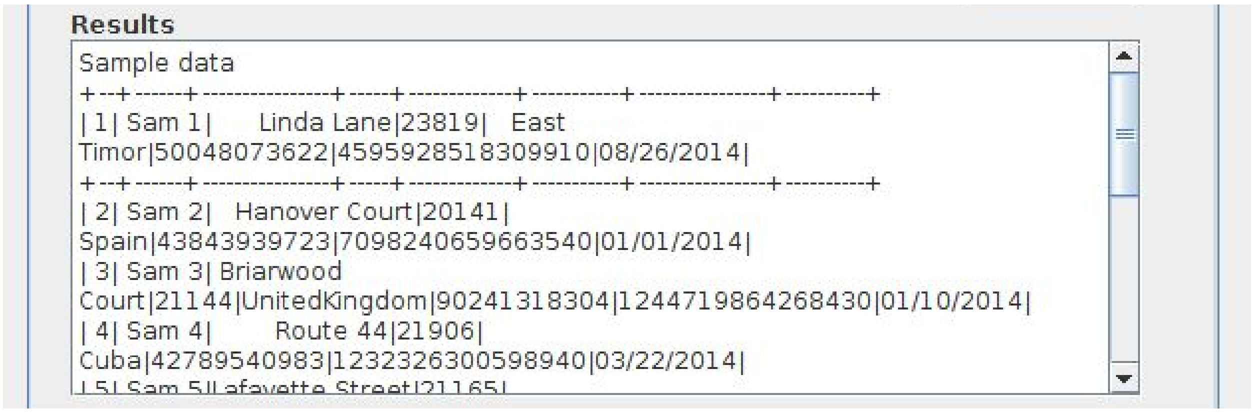
```
select * from Customer_sample
```

File Name: Test .csv

Figure 24

4.4.5 Sample results

Only a small portion of the results is shown inside the Results box, the rests is to be saved to disk. As shown in Figure 25.



Results

```
Sample data
+--+------+---------------+-----+------------+----------+---------------+----------+
| 1| Sam 1|      Linda Lane|23819|   East
Timor|50048073622|4595928518309910|08/26/2014|
+--+------+---------------+-----+------------+----------+---------------+----------+
| 2| Sam 2|   Hanover Court|20141|
Spain|43843939723|7098240659663540|01/01/2014|
| 3| Sam 3| Briarwood
Court|21144|UnitedKingdom|90241318304|1244719864268430|01/10/2014|
| 4| Sam 4|       Route 44|21906|
Cuba|42789540983|1232326300598940|03/22/2014|
| 5| Sam 5|Lafayette Street|21165|
```

Figure 25

5. Possible flows for Data Integration

All possible flows that an execution of one instance are listed here. If one desired task is not listed here, it may be possible to execute it by two executions. For example: Merge and Split, First do a Merge and later do a split job.

5.1 Specifying values manually

1. Input file -> Desired columns -> Output files.
2. Input file -> Desired columns -> Restriction operations -> Output files.
3. Input file -> Desired columns -> Transformation operations [MERGE] -> Output file*.
4. Input file -> Desired columns -> Transformation operations [SPLIT] -> Output file*.
5. Input file -> Desired columns -> Transformation operations [CASING] -> Output files.
6. Input file -> Desired columns -> Transformation operations [CASING] -> Transformation operations [FORMATTING] -> Output files.
7. Input file -> Desired columns -> Transformation operations [CASING] -> Restriction operations -> Output file*.
8. Input file -> Desired columns -> Transformation operations [CASING] -> Transformation operations [FORMATTING] -> Restriction operations -> Output file.
9. Input file -> Desired columns -> Transformation operations [FORMATTING] -> Output file.
10. Input file -> Desired columns -> Transformation operations [FORMATTING] -> Restriction operations -> Output file.
11. Input file -> Desired columns -> Transformation operations [RENAME] -> Output file.
12. Input file -> Desired columns -> Transformation operations [RENAME] -> Restriction operations -> Output file.

5.2 Loading an existing IJSL script

1. Load -> Run.

5.3 Writing a IJSL script

1. Write -> Run.
2. Write -> Save -> Run.

6. Possible flows for Data Analysis

1. List file -> Select files-> Display table names and fields -> Input Query -> Save file -> Display partial results

7. References

[1]http://www.oracle.com/technetwork/database/options/advanced-analytics/bigdataanalyticswpoaa-1930891.pdf

[2] http://harvardmagazine.com/2014/03/why-big-data-is-a-big-deal

[3] http://usc-isi-i2.github.io/papers/knoblock13-sbd.pdf

[4]https://downloads.cloudsecurityalliance.org/initiatives/bdwg/Big_Data_Analytics_for_Security_Intelligence.pdf

[5] https://en.wikipedia.org/wiki/Apache_Hadoop

[6] http://cdn.oreillystatic.com/oreilly/booksamplers/9781449358624_sampler.pdf

[7] http://www.glennklockwood.com/data-intensive/hadoop/mapreduce-workflow.png