User Guide for Calculating Attribute or Value Weight (EXACT Match) in Oyster Scoring Rule by Pig Script

Introduction
Calculate the attribute or value weight used in OYSTER scoring rule to support probabilistic matching. The scoring rule is similar to the Boolean rule in that you can specify a similarity function and optional data preparation function for comparing the values of identity attributes between two entity references. The primary difference is that instead of the similarity resulting in a True or False decision, the decision is whether the identity attribute values should be contribute an "agreement" or "disagreement" weight to an overall match score for the pair of references being compared [1][5].

In the scoring rule, the agree and disagree decisions are both associated with a numerical value call a "weight". Hence, for each identity attribute comparison there is an agreement weight and a disagreement weight. In some cases, there can also be a third weight called a "missing weight" to be used instead of the agreement or disagreement weight in the case either or both values of the identity attribute are missing [2][5].

Depending upon the outcome of the comparison for each identity attribute, either the agreement weight, the disagreement weight, or the missing weight is added into a total score. The total score is then compared to a pre-defined "match score." If the total score is greater than or equal to the match score, then the overall decision is to link the references, otherwise the references are not linked[3][4][5].

An additional feature to support accuracy analysis is the ability to also specify an optional "review score."[2] If a review score is specified, then when the total score falls below the match score, but is greater than or equal to the review score, the pair of references and their total score are written to a "clerical review"[2] file for post-processing analysis[5].

In our Oyster research group, there were two methods to calculate the weight. One is to use Excel spreadsheet, and the other is to use the web version calculator created by R, called OYSTER Weight Calculator. I will use the Pig script to calculate the attribute or value weight in order to automate scoring weight calculations and make the whole process more efficient.

Overview Steps for Calculating Scoring Weights
In order to calculate weights [5], the input file you use must contain

- Record identifier
- Linked identifier (correct linkage [3] identifier or OYSTER identifier, which can be generated by OYSTER Boolean match rules)
- Attributes (e.g. Name, Zip, Address etc.)

Then select the attribute which you want to calculate the weight. In this example, I will select the Name to calculate the weight.

The Weight calculation is followed by the steps below:

1. Calculate total number of possible pairs (Total_Pairs)
2. Calculate number of equivalent pairs (EPairs) and total number of equivalent pairs (Total_EPairs)
3. Calculate the Number of Equivalent Pairs agreeing on Name (AgreeE) and Total Number of Equivalent Pairs agreeing on Name (Total_AgreeE)
4. Calculate Name Frequency (NameFreq), the number of each Name pairs (NamePairs), the number of Name Frequency (NoNameFreq), total number of Name pairs (Total_NamePairs)

5. Calculate Number of Non-Equivalent Pairs agreeing on Name (AgreeNE) = NamePairs - AgreeE
6. Calculate Total Number of Non-Equivalent Pairs agreeing on Name (Total_AgreeNE) = Total_NamePairs - Total_AgreeE
7. Calculate Total Number of Non-Equivalent Pairs (Total_NEPairs) = Total_Pairs - Total_EPairs
8. Calculate Probability of Agreement on each Name for Equivalent Pairs (ProbE) = (float)AgreeE/Total_EPairs
9. Calculate Probability of Agreement on each Name for Non-Equivalent Pairs (ProbNE) = (float)AgreeNE/Total_NEPairs
10. Calculate total probability of agreement on Name for equivalent pairs (Total_ProbE) = (float)Total_AgreeE/Total_EPairs
11. Calculate total probability of agreement on Name for non-equivalent pairs (Total_ProbNE) = (float)Total_AgreeNE/Total_NEPairs
12. Calculate Ratio of Probabilities (Ratio) = (float)ProbE/ProbNE
13. Calculate Agreement Weight for each Name (AgreeWgt) = LOG10(Ratio)/LOG10(2)
14. Calculate Total Ratio of Probabilities (Total_Ratio) = (float)Total_ProbE/Total_ProbNE
15. Calculate Total Agreement Weight for Name (Total_AgreeWgt) = LOG10(Total_Ratio)/LOG10(2)
16. Calculate Disagreement Ratio (CmpRatio) = (float)(1-ProbE)/(1-ProbNE)
17. Calculate Disagreement Weight for each Name (DisAgWgt) = LOG10(CmpRatio)/LOG10(2)
18. Calculate Total Disagreement Ratio (Total_CmpRatio) = (float)(1-Total_ProbE)/(1-Total_ProbNE)
19. Calculate Total Disagreement Weight for Name (Total_DisAgWgt) = LOG10(Total_CmpRatio)/LOG10(2)
20. Calculate the Agreement Weight (MissingAgreeWgt) and Disagreement Weight (MissingDisAgWgt) for Missing values
21. Calculate the agreement weight (AgreeWgt) and disagreement weight (DisAgWgt) for high-frequency attribute values
22. Calculate the Total Agreement Weight (Total_RemnAgreeWgt) and Total Disagreement Weight (Total_RemnDisAgWgt) for Remaining Values. Note: In this project, Oyster research group use the Total disagreement as the Total Disagreement Weight on Remaining Values.

Note: However, if you want to select other attribute to calculate the weight. For example, you want to select Zip and calculate the weight. After step 4-1 (Load file into Pig), replace all the words which include the keyword “Name” to “Zip” in step 4-2 to step 4-9.

Two levels of Scoring Weights

The weight calculation can be done at two levels [5]:

a) At the attribute level, the weight table only has one agreement and disagreement weight per identity attribute (i.e. Total_AgreeWgt and Total_DisAgWgt for Name).
b) At the value-level, there are three weight tables. The first one is a table of agreement and disagreement weights for high-frequency identity attribute values (AgreeWgt and DisAgWgt for each Name). The second one is a table contains Agreement Weight and Disagreement Weight for Missing values (MissingAgreeWgt and MissingDisAgWgt). The third one is a table of Total Agreement Weight and Total Disagreement Weight for Remaining Values (Total_RemnAgreeWgt and Total_RemnDisAgWgt for remaining Names).

Weight Calculation Step by Step

In this example, the file I use is a synthetic data 'Missing values Test File.txt', it contains RecID (Record identifier), Name (attribute), ClusterID (Linked identifier). After preparation, I saved it as 'Missing values Test File_blank to NA_no header.csv' and use it as an input file.

A: Data Preparation

1. **Change Missing values**

   Replace all missing values to a specified value. If the type of attribute is a letter, replace all missing values to NA, and the condition is that there is no NA exist in the specified column(s); if the type of attribute is an integer, replace all missing values to 0, and the condition is that there is no 0 exist in the specified column(s). For this example, I replaced all missing values in Name column to NA and replaced all missing values in Zip column to 0.

   - In Pig the concept of null is the same as in SQL [6]. If one value is null, then it is neither equals nor not equals. When comparing datasets, don't forget to check whether the value is empty or not, or we might miss some data.

2. **Change Case**

   Convert all letters to Proper Case or Upper Case or Lower Case (convert to the consistent case).

   - The names (aliases) of relations and fields are case sensitive [7] in Pig. For example, there are two same names, but one is upper case, another is lower case, Pig will treat these two as different names.

3. **Remove Header**

   - Since in Pig each line is a separate record of data [8]. If header is Name, ClusterID, and so unless there is really a person named Name, with a ClusterID of ClusterID, having such a line is wrong. Pig makes no guarantees about the order in which fields will be output (unless using ORDER BY), so header row might show up anywhere.

```
RecID,Name,ClusterID
A001,James,ZY1
A002,Mary,KF6
A003,,TK8
A004,Jim,ZY1
A005,William,DR4
A006,William,TK8
A007,Mary,KF6
A008,James,WD5
A009,Harry,NJ8
A010,Marie,KF6
A011,James,WD5
A012,,DR4
A013,Harold,NJ8
A014,Mary,KF6
A015,Jim,WD5
A016,James,ZY1
A017,Mary,KF6
A018,Marie,MF2
A019,William,TK8
A020,,DR4
A021,James,WD5
A022,Marie,MF2
A023,William,TK8
A024,James,WD5
A025,James,ZY1
```

Before Preparation

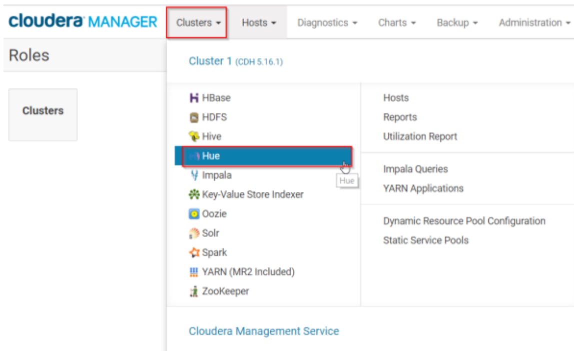
```
A001,James,ZY1
A002,Mary,KF6
A003,NA,TK8
A004,Jim,ZY1
A005,William,DR4
A006,William,TK8
A007,Mary,KF6
A008,James,WD5
A009,Harry,NJ8
A010,Marie,KF6
A011,James,WD5
A012,NA,DR4
A013,Harold,NJ8
A014,Mary,KF6
A015,Jim,WD5
A016,James,ZY1
A017,Mary,KF6
A018,Marie,MF2
A019,William,TK8
A020,NA,DR4
A021,James,WD5
A022,Marie,MF2
A023,William,TK8
A024,James,WD5
A025,James,ZY1
```

After Preparation

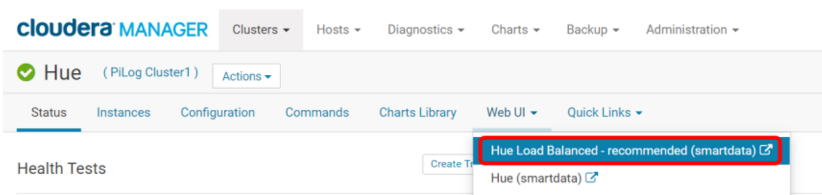
B: Upload the Input File(s) to HDFS through Hue

Step 1: Login Hue

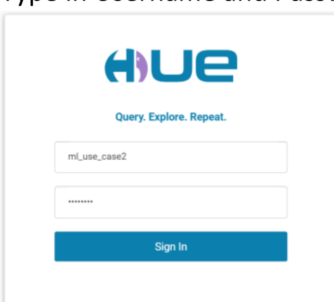
1. Log on to Cloudera Manager - Please see < Cloudera User Guide - B: Accessing Cloudera through Cloudera Manager>
2. Navigate to clusters and select Hue



3. Access the Hue Web UI

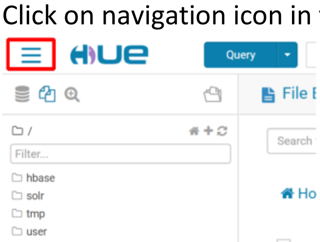


4. Type in Username and Password when prompted: ml_use_case2 P2tnH&mL

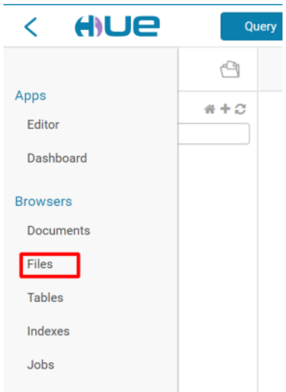


Step 2: Creating Directory in HDFS

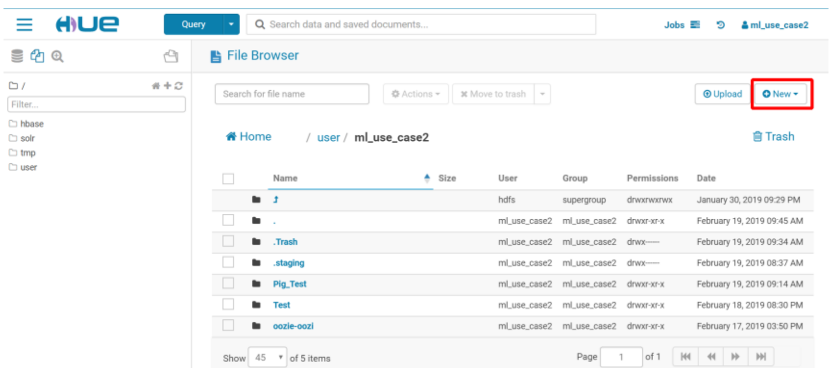
1. Click on navigation icon in the top left corner



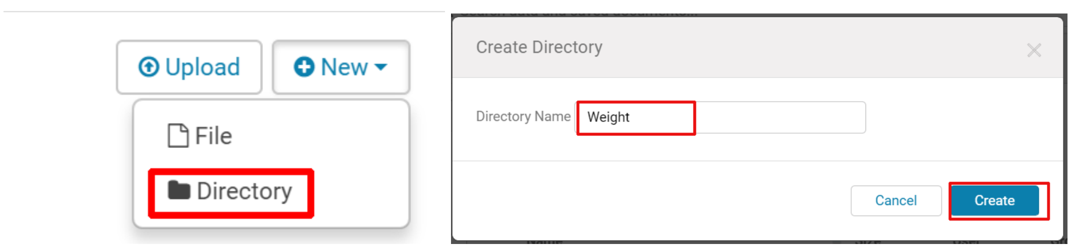
2. Click on Files



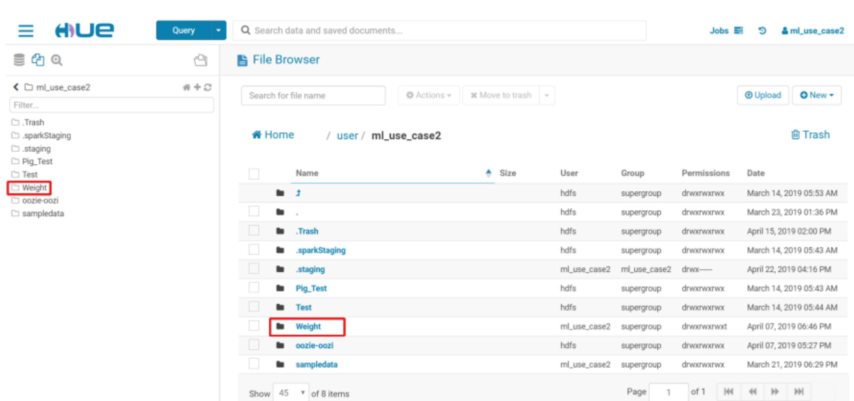
3. Once you have clicked on the Files all the Directory will showing, then click on "+ new" in the top right corner



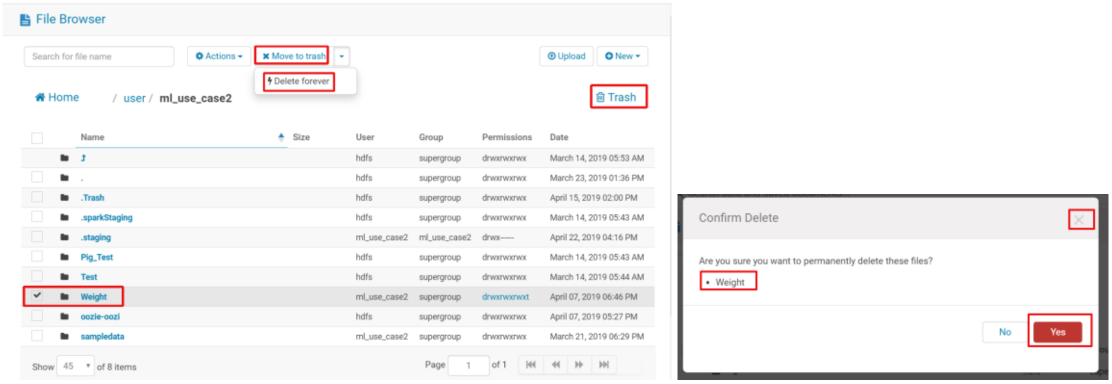
4. Click on Directory. Name your Directory (Note: No spaces are allowed), click on Create, wait for more than 5 seconds then close it, then the new directory will be created



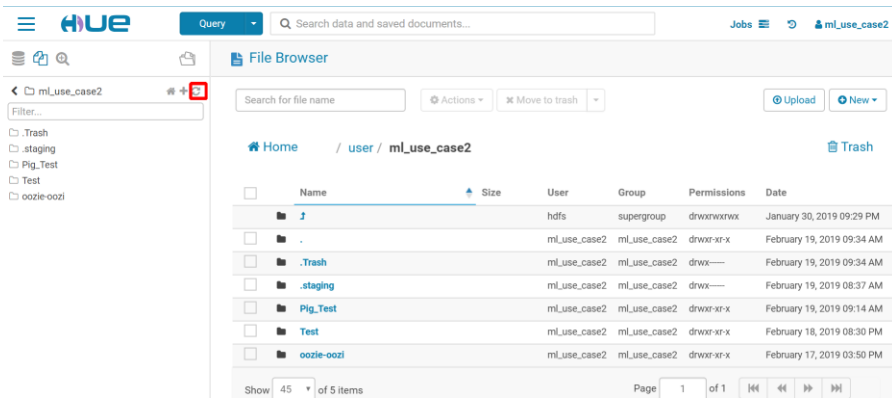
5. Refresh the web and click on refresh icon, you can see new Directory



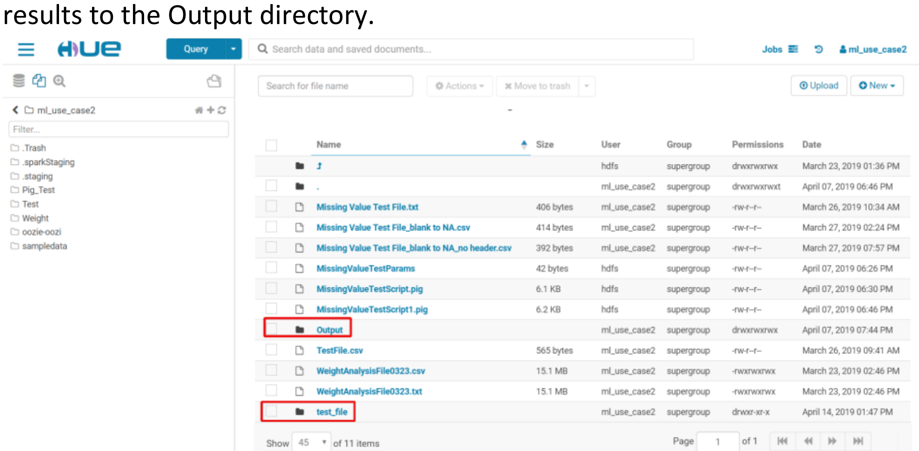
6. If you want to delete it, select it, click on "× Move to trash", you can see it in trash (click Trash in the top right corner, you will see it after you done), or you can Delete forever, then click on "Yes", waiting for more than 5 seconds then close it



7. Refresh web and click on refresh icon, you can see the Directory you chosen have been deleted



8. Use the same method to create two new Directories (test_file Directory, Output Directory) under the Weight Directory, you can upload the files to the test_file Directory and save the results to the Output directory.



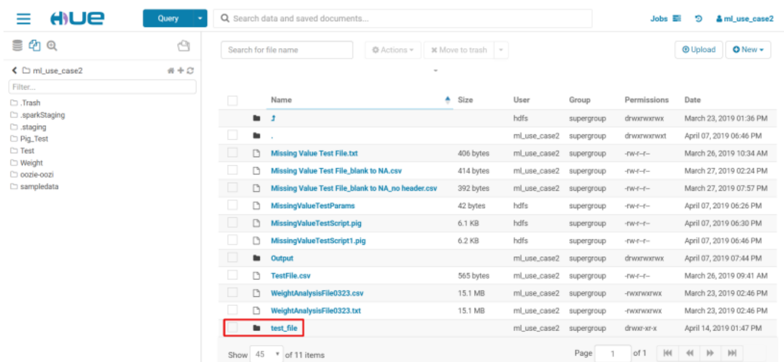
Step 3: Upload file(s) to HDFS

Please note that there are 2 methods to upload file(s) to HDFS.

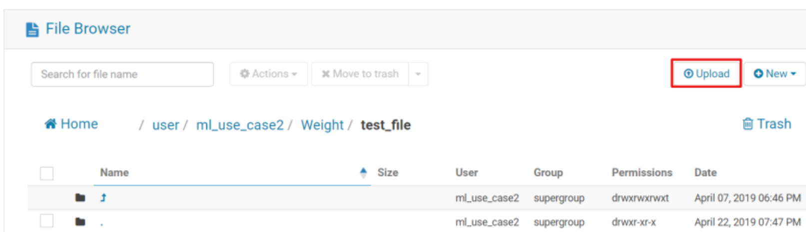
Method 1:

Note: The user has the option to either drag and drop the specific file from their machine into the directory or to search for the specific file on their machine by selecting the browser and selecting "Upload":

1. Click on "test_file" you just created

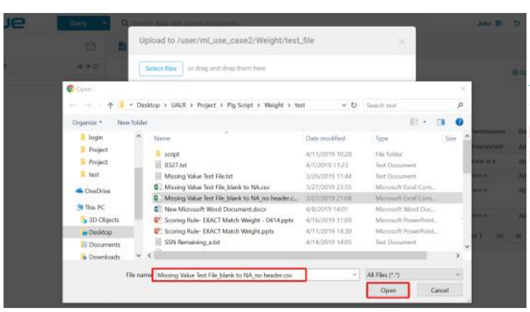


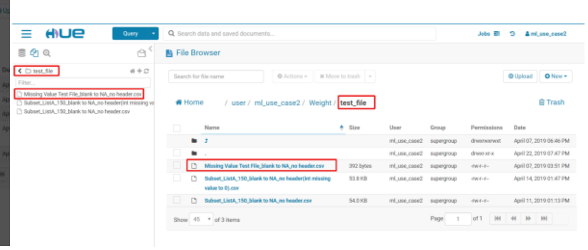
2. Click on "Upload" in the top right corner (or Drag and drop the specific file from their machine into the directory)



3. Click on "Select files"

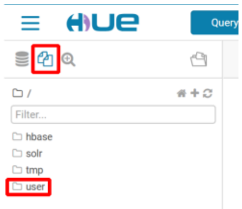
4. Select the file(s) which is located on your local machine that you want to be uploaded, click on "Open", you will see the file under test_file directory



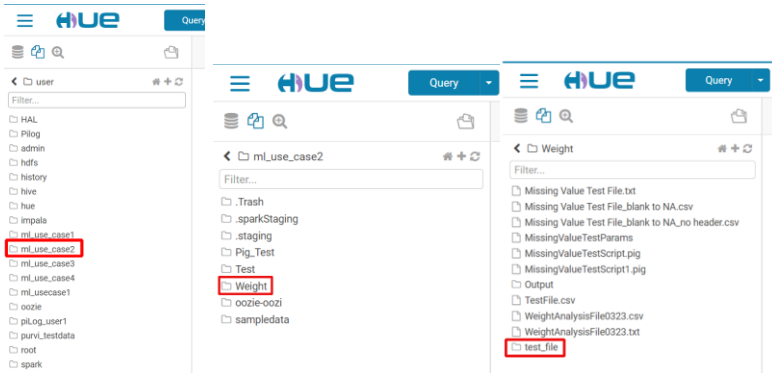


Method 2:

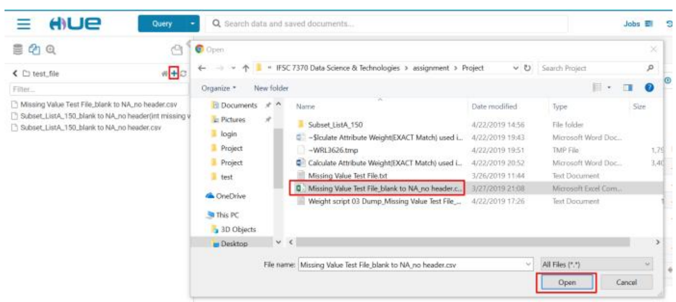
1. Click on HDFS icon in the top left corner, then select user



2. Click on ml_use_case2, then select Weight, then select test_file



3. Click on "+" icon in the top right corner, select the file(s) which is located on your local machine that you want to be uploaded, click on "Open", you will see the file under test_file directory



C: Set Up PuTTY Environment and Run Pig in Grunt Shell

Step 1: Download and Login PuTTY

Please see < Cloudera User Guide - A: Accessing Cloudera through PuTTY>

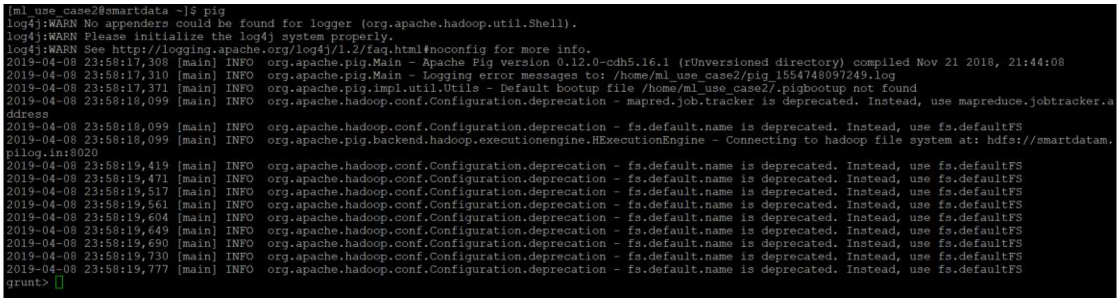
Step 2: Set hadoop user name to hdfs

The command: export HADOOP_USER_NAME=hdfs

```
login as: ml_use_case2
ml_use_case2@smartdata.pilog.in's password:
Last login: Sun Apr  7 21:21:16 2019 from 99.64.177.178
[ml_use_case2@smartdata ~]$ export HADOOP_USER_NAME=hdfs
```

Step 3: Get into Pig

Type in Pig, get into the MapReduce mode.


```
[ml_use_case2@smartdata ~]$ pig
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2019-04-08 23:58:17,308 [main] INFO  org.apache.pig.Main - Apache Pig version 0.12.0-cdh5.16.1 (rUnversioned directory) compiled Nov 21 2018, 21:44:08
2019-04-08 23:58:17,310 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/ml_use_case2/pig_1554748097249.log
2019-04-08 23:58:17,371 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file /home/ml_use_case2/.pigbootup not found
2019-04-08 23:58:18,099 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.a
ddress
2019-04-08 23:58:18,099 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2019-04-08 23:58:18,099 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://smartdatam.
pilog.in:8020
2019-04-08 23:58:19,419 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2019-04-08 23:58:19,471 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2019-04-08 23:58:19,517 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2019-04-08 23:58:19,561 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2019-04-08 23:58:19,604 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2019-04-08 23:58:19,649 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2019-04-08 23:58:19,690 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2019-04-08 23:58:19,730 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2019-04-08 23:58:19,777 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt>
```

Step 4: Run Pig in Grunt shell

Run Pig using the Grunt shell. Exit Pig by pressing "Ctrl+D".

Weight Calculation with Pig script step by step:

Note: All the code has been listed in <Weight Calculator Pig Script.txt>. You just need to copy the code to the Grunt Shell and follow by the steps, then you will get the result. Remember that in your script, you can comment out any commands by coding two dashes (--) before the command.

grunt>

1. **Load file into Pig**
   1.1. Use the LOAD operator and the load functions to read the data from a comma separated values file into Pig.
      1.1.1. Syntax: LOAD 'data' [USING function] [AS schema][7][9][10]. For example: load '/user/ml_use_case2/Weight/Missing values Test File_blank to NA_no header.csv' using PigStorage(',') as (RecID:chararray, Name:chararray, Zip:int, ClusterID:chararray).
         1.1.1.1. Read the data from /user/ml_use_case2/Weight/Missing values Test File_blank to NA_no header.csv into a relation called WeightAnalysisFile using the default PigStorage() loader.
         1.1.1.2. Specify a comma as being the separating character and produce the data of the type specified by the schema: AS (RecID:chararray, Name:chararray, ClusterID:chararray).
   1.2. Then dump [7] your WeightAnalysisFile file the following command.
      1.2.1. Use DUMP statement to display results to your terminal screen (You can use the up and down cursor keys to recall previous command).
      1.2.2. If you only want to see the final result (step 7: Total_AgreeWgt, total_DisAgWgt, step 8: MissingAgreeWgt, MissingDisAgWgt, step 9: AgreeWgt, DisAgWgt, Total_RemnAgreeWgt, Total_RemnDisAgWgt), you just need to delete dump code form step 1- 6.

```
WeightAnalysisFile = load '/user/ml_use_case2/Weight/Missing values
Test File_blank to NA_no header.csv' using PigStorage(',') as
(RecID:chararray, Name:chararray, ClusterID:chararray);

dump WeightAnalysisFile;
```

```
(A001,James,ZY1)
(A002,Mary,KF6)
(A003,NA,TK8)
(A004,Jim,ZY1)
(A005,William,DR4)
(A006,William,TK8)
(A007,Mary,KF6)
(A008,James,WD5)
(A009,Harry,NJ8)
(A010,Marie,KF6)
(A011,James,WD5)
(A012,NA,DR4)
(A013,Harold,NJ8)
(A014,Mary,KF6)
(A015,Jim,WD5)
(A016,James,ZY1)
(A017,Mary,KF6)
(A018,Marie,MF2)
(A019,William,TK8)
(A020,NA,DR4)
(A021,James,WD5)
(A022,Marie,MF2)
(A023,William,TK8)
(A024,James,WD5)
(A025,James,ZY1)
```

2. Calculate Total_Pairs

2.1. Group WeightAnalysisFile all. Use GROUP ALL [7] to group all to go to a single group. You can think of 'all' as the group name, and think of other fields (the bag) as an inner bag.

```
WeightAnalysisFile_group = GROUP WeightAnalysisFile ALL;

dump WeightAnalysisFile_group;
```

```
(all,{(A025,James,ZY1),(A024,James,WD5),(A023,William,TK8),(A022,Marie,MF2),(A021,James,WD5),(A020,NA,DR4),(A019,William,TK8),(A018,Marie,MF2),(A017,Mary,KF6
),(A016,James,ZY1),(A015,Jim,WD5),(A014,Mary,KF6),(A013,Harold,NJ8),(A012,NA,DR4),(A011,James,WD5),(A010,Marie,KF6),(A009,Harry,NJ8),(A008,James,WD5),(A007,M
ary,KF6),(A006,William,TK8),(A005,William,DR4),(A004,Jim,ZY1),(A003,NA,TK8),(A002,Mary,KF6),(A001,James,ZY1)})
```

2.2. Use FOREACH [7] operator to generate the relation only have ClusterID and total number of records (PairsCount). The total number of records is generated by counting the total number of ClusterID using COUNT function [7][11][12].

Note: COUNT requires a preceding GROUP ALL statement for global counts and a GROUP BY statement for group counts.

So I use GROUP ALL statement in the previous step in order to COUNT the total number of ClusterID.

```
Pairs_Count = FOREACH WeightAnalysisFile_group Generate
WeightAnalysisFile.ClusterID as ClusterID,
COUNT(WeightAnalysisFile.ClusterID) as PairsCount;

dump Pairs_Count;
```

```
({(ZY1),(WD5),(TK8),(MF2),(WD5),(DR4),(TK8),(MF2),(KF6),(ZY1),(WD5),(KF6),(NJ8),(DR4),(WD5),(KF6),(NJ8),(WD5),(KF6),(TK8),(DR4),(ZY1),(TK8),(KF6),(ZY1)},25)
```

2.3. Use Nested FOREACH...GENERATE block with an inner bag[7] to generate the relation only have ClusterID and Total_Pairs. Total_Pairs is performed within the nested block[7] by using n*(n-1)/2 (e.g. total number of records * (total number of records – 1)/ 2).

```
Total_Pairs_Count = FOREACH WeightAnalysisFile_group {Total_Pairs =
Pairs_Count.PairsCount * (Pairs_Count.PairsCount - 1) / 2; Generate
WeightAnalysisFile.ClusterID as ClusterID, Total_Pairs as
Total_Pairs;};
```

```
dump Total_Pairs_Count;
```

```
({(ZY1),(WD5),(TK8),(MF2),(WD5),(DR4),(TK8),(MF2),(KF6),(ZY1),(WD5),(KF6),(NJ8),(DR4),(WD5),(KF6),(NJ8),(WD5),(KF6),(TK8),(DR4),(ZY1),(TK8),(KF6),(ZY1)},300)
```

3. Calculate EPairs, Total_EPairs

3.1. Use FOREACH operator to generate the relation only have ClusterID.

```
EPairs_Count_a = FOREACH WeightAnalysisFile GENERATE ClusterID;

dump EPairs_Count_a;
```

```
(ZY1)
(KF6)
(TK8)
(ZY1)
(DR4)
(TK8)
(KF6)
(WD5)
(NJ8)
(KF6)
(WD5)
(DR4)
(NJ8)
(KF6)
(WD5)
(ZY1)
(KF6)
(MF2)
(TK8)
(DR4)
(WD5)
(MF2)
(TK8)
(WD5)
(ZY1)
```

3.2. Group it by ClusterID.

```
EPairs_Count_a_group = GROUP EPairs_Count_a by ClusterID;

dump EPairs_Count_a_group;
```

```
(DR4,{(DR4),(DR4),(DR4)})
(KF6,{(KF6),(KF6),(KF6),(KF6),(KF6)})
(MF2,{(MF2),(MF2)})
(NJ8,{(NJ8),(NJ8)})
(TK8,{(TK8),(TK8),(TK8),(TK8)})
(WD5,{(WD5),(WD5),(WD5),(WD5),(WD5)})
(ZY1,{(ZY1),(ZY1),(ZY1),(ZY1)})
```

3.3. Use FOREACH operator from the GROUP operator to generate the relation only have ClusterID and the number of each ClusterID. The number of each ClusterID is generated by counting the number of each ClusterID using the COUNT function.

Before this step, I use GROUP BY ClusterID in order to COUNT the number of each ClusterID.

```
EPairs_Count_b = FOREACH EPairs_Count_a_group GENERATE group as
ClusterID, COUNT($1) as count;

dump EPairs_Count_b;
```

```
(DR4,3)
(KF6,5)
(MF2,2)
(NJ8,2)
(TK8,4)
(WD5,5)
(ZY1,4)
```

3.4. Use FOREACH operator on EPairs_Count_b to generate the relation only have ClusterID and the number of equivalent pairs based on the ClusterID size (EPairs). The number of equivalent pairs generated by using n*(n-1)/2 (e.g. number of each ClusterID * (number of each ClusterID – 1)/2).

```
EPairs_Count_c = FOREACH EPairs_Count_b {EPairs = count * (count - 1) / 2; GENERATE ClusterID as ClusterID, EPairs as EPairs;};

dump EPairs_Count_c;
```

```
(DR4,3)
(KF6,10)
(MF2,1)
(NJ8,1)
(TK8,6)
(WD5,10)
(ZY1,6)
```

3.5. Group EPairs_Count_c all.

```
EPairs_Count_c_group = GROUP EPairs_Count_c ALL;

dump EPairs_Count_c_group;
```

```
(all,{(ZY1,6),(WD5,10),(TK8,6),(NJ8,1),(MF2,1),(KF6,10),(DR4,3)})
```

3.6. Use FOREACH operator to generate the relation only have ClusterID and Total_EPairs, The total number of equivalent pairs is generated by summing the number of equivalent pairs using SUM function[7][11][12].

Note: SUM also requires a preceding GROUP ALL statement for global sums and a GROUP BY statement for group sums.

```
Total_EPairs_Count = FOREACH EPairs_Count_c_group Generate EPairs_Count_c.ClusterID as ClusterID, SUM(EPairs_Count_c.EPairs) as Total_EPairs;

dump Total_EPairs_Count;
```

```
({(ZY1),(WD5),(TK8),(NJ8),(MF2),(KF6),(DR4)},37)
```

4. Calculate AgreeE, Total_AgreeE

4.1. Group the WeightAnalysisFile having the same Name and same ClusterID.

```
AgreeE_Count_a_group = GROUP WeightAnalysisFile by (Name, ClusterID);
```

```
dump AgreeE_Count_a_group;
```

```
((NA,DR4),{(A020,NA,DR4),(A012,NA,DR4)})
((NA,TK8),{(A003,NA,TK8)})
((Jim,WD5),{(A015,Jim,WD5)})
((Jim,ZY1),{(A004,Jim,ZY1)})
((Mary,KF6),{(A007,Mary,KF6),(A017,Mary,KF6),(A014,Mary,KF6),(A002,Mary,KF6)})
((Harry,NJ8),{(A009,Harry,NJ8)})
((James,WD5),{(A024,James,WD5),(A008,James,WD5),(A021,James,WD5),(A011,James,WD5)})
((James,ZY1),{(A001,James,ZY1),(A016,James,ZY1),(A025,James,ZY1)})
((Marie,KF6),{(A010,Marie,KF6)})
((Marie,MF2),{(A018,Marie,MF2),(A022,Marie,MF2)})
((Harold,NJ8),{(A013,Harold,NJ8)})
((William,DR4),{(A005,William,DR4)})
((William,TK8),{(A006,William,TK8),(A019,William,TK8),(A023,William,TK8)})
```

4.2. Use FOREACH operator to generate the relation only have ClusterID and number of records having the same Name and same ClusterID. The number of records having the same Name and same ClusterID is generated by using COUNT function.

```
AgreeE_Count_b = FOREACH AgreeE_Count_a_group GENERATE group.Name as Name, group.ClusterID as ClusterID, COUNT($1) as AgreeECount_b;

dump AgreeE_Count_b;
```

```
(NA,DR4,2)
(NA,TK8,1)
(Jim,WD5,1)
(Jim,ZY1,1)
(Mary,KF6,4)
(Harry,NJ8,1)
(James,WD5,4)
(James,ZY1,3)
(Marie,KF6,1)
(Marie,MF2,2)
(Harold,NJ8,1)
(William,DR4,1)
(William,TK8,3)
```

4.3. Use FOREACH operator to generate the relation only have Name, ClusterID and the number of equivalent pairs agreeing on each Name for each subgroup[5] (Subgroup refers to all groups with the same Name and the same ClusterID for a standardized value (Name). For example, NA have two subgroups, they are (NA,DR4,1) and (NA,TK8,0)). The number of equivalent pairs agreeing on each Name for each subgroup is generated by using n*(n-1)/2 (e.g. number of records having the same Name and same ClusterID * (number of records having the same Name and same ClusterID – 1)/ 2).

```
AgreeE_Count_c = FOREACH AgreeE_Count_b {AgreeE_c = AgreeECount_b * (AgreeECount_b - 1) / 2; GENERATE Name as Name, ClusterID as ClusterID, AgreeE_c as AgreeE_c;};

dump AgreeE_Count_c;
```

```
(NA,DR4,1)
(NA,TK8,0)
(Jim,WD5,0)
(Jim,ZY1,0)
(Mary,KF6,6)
(Harry,NJ8,0)
(James,WD5,6)
(James,ZY1,3)
(Marie,KF6,0)
(Marie,MF2,1)
(Harold,NJ8,0)
(William,DR4,0)
(William,TK8,3)
```

4.4. FOREACH AgreeE_Count_c generate the relation only have Name and the number of equivalent pairs agreeing on each Name for each subgroup (AgreeE_c).

```
AgreeE_Count_d = FOREACH AgreeE_Count_c generate Name as Name,
AgreeE_c as AgreeE_c;
```

```
dump AgreeE_Count_d;
```

```
(NA,1)
(NA,0)
(Jim,0)
(Jim,0)
(Mary,6)
(Harry,0)
(James,6)
(James,3)
(Marie,0)
(Marie,1)
(Harold,0)
(William,0)
(William,3)
```

4.5. Group AgreeE_Count_d by Name.

```
AgreeE_Distinct_Count = GROUP AgreeE_Count_d by Name;
```

```
dump AgreeE_Distinct_Count;
```

```
(NA,{(NA,1),(NA,0)})
(Jim,{(Jim,0),(Jim,0)})
(Mary,{(Mary,6)})
(Harry,{(Harry,0)})
(James,{(James,6),(James,3)})
(Marie,{(Marie,0),(Marie,1)})
(Harold,{(Harold,0)})
(William,{(William,0),(William,3)})
```

4.6. FOREACH AgreeE_Distinct_Count generate the relation only have distinct Name and AgreeE, AgreeE is generated by summing the equivalent pairs for each Name for all subgroups using SUM function.

```
AgreeE_Distinct = FOREACH AgreeE_Distinct_Count GENERATE group as
Name, SUM(AgreeE_Count_d.AgreeE_c) as AgreeE;
```

```
dump AgreeE_Distinct;
```

```
(NA,1)
(Jim,0)
(Mary,6)
(Harry,0)
(James,9)
(Marie,1)
(Harold,0)
(William,3)
```

4.7. Group AgreeE_Distinct all.

```
AgreeE_Distinct_Count_group = GROUP AgreeE_Distinct all;
```

```
dump AgreeE_Distinct_Count_group;
```

```
(all,{(William,3),(Harold,0),(Marie,1),(James,9),(Harry,0),(Mary,6),(Jim,0),(NA,1)})
```

4.8. FOREACH the GROUP generate the relation only have Name and Total_AgreeE. Total_AgreeE is generated by summing the number of equivalent pairs agreeing on each Name using SUM function.

```
Total_AgreeE_Count = FOREACH AgreeE_Distinct_Count_group Generate
AgreeE_Distinct.Name as Name, SUM(AgreeE_Distinct.AgreeE) as
Total_AgreeE;
```

```
dump Total_AgreeE_Count;
```

```
({(William),(Harold),(Marie),(James),(Harry),(Mary),(Jim),(NA)},20)
```

5. Calculate NameFreq, NamePairs, NoNameFreq, Total_NamePairs

5.1. Use FOREACH operator to generate the relation only have Name.

```
NamePairs_Count_a = FOREACH WeightAnalysisFile GENERATE Name;
```

```
dump NamePairs_Count_a;
```

```
(James)
(Mary)
(NA)
(Jim)
(William)
(William)
(Mary)
(James)
(Harry)
(Marie)
(James)
(NA)
(Harold)
(Mary)
(Jim)
(James)
(Mary)
(Marie)
(William)
(NA)
(James)
(Marie)
(William)
(James)
(James)
```

5.2. Group it by Name.

```
NamePairs_Count_a_group = GROUP NamePairs_Count_a by Name;
```

```
dump NamePairs_Count_a_group;
```

```
(NA,{(NA),(NA),(NA)})
(Jim,{(Jim),(Jim)})
(Mary,{(Mary),(Mary),(Mary),(Mary)})
(Harry,{(Harry)})
(James,{(James),(James),(James),(James),(James),(James),(James)})
(Marie,{(Marie),(Marie),(Marie)})
(Harold,{(Harold)})
(William,{(William),(William),(William),(William)})
```

5.3. Use FOREACH operator to generate the relation only have Name and NameFreq. Name Frequency is generated by counting the number of each Name using COUNT function.

```
NameFreq_Count = FOREACH NamePairs_Count_a_group GENERATE group as
Name, COUNT($1) as NameFreq;
```

```
dump NameFreq_Count;
```

```
(NA,3)
(Jim,2)
(Mary,4)
(Harry,1)
(James,7)
(Marie,3)
(Harold,1)
(William,4)
```

5.4. Use FOREACH operator to generate the relation only have Name and NamePairs. NamePairs is generated by using n * (n – 1)/ 2 (e.g. NameFreq * (NameFreq - 1) / 2).

```
NamePairs_Count = FOREACH NameFreq_Count {NamePairs = NameFreq *
(NameFreq - 1) / 2; GENERATE Name as Name, NamePairs as NamePairs;};
```

```
dump NamePairs_Count;
```

```
(NA,3)
(Jim,1)
(Mary,6)
(Harry,0)
(James,21)
(Marie,3)
(Harold,0)
(William,6)
```

5.5. Group it all.

```
NamePairs_Count_group = GROUP NamePairs_Count ALL;
```

```
dump NamePairs_Count_group;
```

```
(all,{(William,6),(Harold,0),(Marie,3),(James,21),(Harry,0),(Mary,6),(Jim,1),(NA,3)})
```

5.6. Use FOREACH operator to generate the relation only have Name, NoNameFreq and Total_NamePairs. NoNameFreq is generated by counting the total number of unique Name using COUNT function, Total_NamePairs is generated by summing NamePairs using SUM function.

```
Total_NamePairs_Count = FOREACH NamePairs_Count_group GENERATE NamePairs_Count.Name as Name, COUNT(NamePairs_Count.Name) as NoNameFreq, SUM(NamePairs_Count.NamePairs) as Total_NamePairs;

dump Total_NamePairs_Count;
```

```
({(William),(Harold),(Marie),(James),(Harry),(Mary),(Jim),(NA)},8,40)
```

6. Generate overall table contains all needed values

Contains: Name, NameFreq, NoNameFreq, AgreeE, NamePairs, AgreeNE, Total_Pairs, Total_EPairs, Total_AgreeE, Total_NamePairs, Total_AgreeNE, Total_NEPairs, ProbE, ProbNE, Total_ProbE, Total_ProbNE, Ratio, AgreeWgt, CmpRatio, DisAgWgt, Total_Ratio, Total_AgreeWgt, Total_CmpRatio, Total_DisAgWgt

6.1. Start by joining[7] the AgreeE_Distinct relation, the NameFreq_Count relation and the NamePairs_Count relation on Name.

```
Table_1 = join AgreeE_Distinct by Name, NameFreq_Count by Name, NamePairs_Count by Name;

dump Table_1;
```

```
(NA,1,NA,3,NA,3)
(Jim,0,Jim,2,Jim,1)
(Mary,6,Mary,4,Mary,6)
(Harry,0,Harry,1,Harry,0)
(James,9,James,7,James,21)
(Marie,1,Marie,3,Marie,3)
(Harold,0,Harold,1,Harold,0)
(William,3,William,4,William,6)
```

6.2. Next project a new relation so that you only are working with the distinct Name and the AgreeE of the AgreeE_Distinct, the NameFreq of the NameFreq_Count and the NamePairs of the NamePairs_Count.

```
Table_2 = FOREACH Table_1 GENERATE AgreeE_Distinct::Name as Name, NameFreq_Count::NameFreq as NameFreq, AgreeE_Distinct::AgreeE as AgreeE, NamePairs_Count::NamePairs as NamePairs;

dump Table_2;
```

```
(NA,3,1,3)
(Jim,2,0,1)
(Mary,4,6,6)
(Harry,1,0,0)
(James,7,9,21)
(Marie,3,1,3)
(Harold,1,0,0)
(William,4,3,6)
```

6.3. Use the FOREACH operator to generate the relation only have distinct Name, NameFreq, NoNameFreq, AgreeE, NamePairs, AgreeNE, Total_EPairs, Total_AgreeE, and Total_NamePairs. AgreeNE is performed within the nested block by using NamePairs – AgreeE.

```
Table_3 = FOREACH Table_2 {AgreeNE = NamePairs - AgreeE; GENERATE
Name as Name, NameFreq as NameFreq, Total_NamePairs_Count.NoNameFreq
as NoNameFreq, AgreeE as AgreeE, NamePairs as NamePairs, AgreeNE as
AgreeNE, Total_Pairs_Count.Total_Pairs as Total_Pairs,
Total_EPairs_Count.Total_EPairs as Total_EPairs,
Total_AgreeE_Count.Total_AgreeE as Total_AgreeE,
Total_NamePairs_Count.Total_NamePairs as Total_NamePairs;};
```

```
dump Table_3;
```

```
(NA,3,8,1,3,2,300,37,20,40)
(Jim,2,8,0,1,1,300,37,20,40)
(Mary,4,8,6,6,0,300,37,20,40)
(Harry,1,8,0,0,0,300,37,20,40)
(James,7,8,9,21,12,300,37,20,40)
(Marie,3,8,1,3,2,300,37,20,40)
(Harold,1,8,0,0,0,300,37,20,40)
(William,4,8,3,6,3,300,37,20,40)
```

6.4. Use the FOREACH operator, Total_AgreeNE is performed within the nested block by using Total_NamePairs - Total_AgreeE, Total_NEPairs is performed within the nested block by using Total_Pairs - Total_EPairs, and then generate Table_4.

```
Table_4 = FOREACH Table_3 {Total_AgreeNE = Total_NamePairs -
Total_AgreeE; Total_NEPairs = Total_Pairs - Total_EPairs; GENERATE
Name as Name, NameFreq as NameFreq, NoNameFreq as NoNameFreq, AgreeE
as AgreeE, NamePairs as NamePairs, AgreeNE as AgreeNE, Total_Pairs
as Total_Pairs, Total_EPairs as Total_EPairs, Total_AgreeE as
Total_AgreeE, Total_NamePairs as Total_NamePairs, Total_AgreeNE as
Total_AgreeNE, Total_NEPairs as Total_NEPairs;};
```

```
dump Table_4;
```

```
(NA,3,8,1,3,2,300,37,20,40,20,263)
(Jim,2,8,0,1,1,300,37,20,40,20,263)
(Mary,4,8,6,6,0,300,37,20,40,20,263)
(Harry,1,8,0,0,0,300,37,20,40,20,263)
(James,7,8,9,21,12,300,37,20,40,20,263)
(Marie,3,8,1,3,2,300,37,20,40,20,263)
(Harold,1,8,0,0,0,300,37,20,40,20,263)
(William,4,8,3,6,3,300,37,20,40,20,263)
```

6.5. Use the FOREACH operator, ProbE, ProbNE, Total_ProbE and Total_ProbNE are performed within the nested block, and then generate Table_5.

Note: To divide two integers to get a float number, first convert the integer to a float number using FLOAT and then calculate; When calculating probabilities, use 0.001 instead of zero, by using bincond operator[7], the Symbol is ? : (condition ? value_if_true : value_if_false, e.g. ProbE

== 0.0 ? 0.001:ProbE, in this example, the condition is: ProbE == 0.0, if it is true, return 0.001, if it is false, return ProbE).

```
Table_5 = FOREACH Table_4 {ProbE = (float)AgreeE/Total_EPairs;
ProbNE = (float)AgreeNE/Total_NEPairs; Total_ProbE =
(float)Total_AgreeE/Total_EPairs; Total_ProbNE =
(float)Total_AgreeNE/Total_NEPairs; GENERATE Name as Name, NameFreq
as NameFreq, NoNameFreq as NoNameFreq, AgreeE as AgreeE, NamePairs
as NamePairs, AgreeNE as AgreeNE, Total_Pairs as Total_Pairs,
Total_EPairs as Total_EPairs, Total_AgreeE as Total_AgreeE,
Total_NamePairs as Total_NamePairs, Total_AgreeNE as Total_AgreeNE,
Total_NEPairs as Total_NEPairs, (ProbE == 0.0 ? 0.001:ProbE) as
ProbE, (ProbNE == 0.0 ? 0.001:ProbNE) as ProbNE, Total_ProbE as
Total_ProbE, Total_ProbNE as Total_ProbNE;};
```

```
dump Table_5;
```

```
(NA,3,8,1,3,2,300,37,20,40,20,263,0.027027027681469917,0.0076045626774430275,0.5405405,0.076045625)
(Jim,2,8,0,1,1,300,37,20,40,20,263,0.001,0.0038022813387215137,0.5405405,0.076045625)
(Mary,4,8,6,6,0,300,37,20,40,20,263,0.1621621549129486,0.001,0.5405405,0.076045625)
(Harry,1,8,0,0,0,300,37,20,40,20,263,0.001,0.001,0.5405405,0.076045625)
(James,7,8,9,21,12,300,37,20,40,20,263,0.2432432472705841,0.045627377927303314,0.5405405,0.076045625)
(Marie,3,8,1,3,2,300,37,20,40,20,263,0.027027027681469917,0.0076045626774430275,0.5405405,0.076045625)
(Harold,1,8,0,0,0,300,37,20,40,20,263,0.001,0.001,0.5405405,0.076045625)
(William,4,8,3,6,3,300,37,20,40,20,263,0.0810810774564743,0.011406844481825829,0.5405405,0.076045625)
```

6.6. Use the FOREACH operator, Ratio, AgreeWgt, CmpRatio, DisAgWgt, Total_Ratio, Total_AgreeWgt, Total_CmpRatio, Total_DisAgWgt are performed within the nested block, and then generate Table_6.

```
Table_6 = FOREACH Table_5 {Ratio = (float)ProbE/ProbNE; AgreeWgt =
LOG10(Ratio)/LOG10(2); CmpRatio = (float)(1-ProbE)/(1-ProbNE);
DisAgWgt = LOG10(CmpRatio)/LOG10(2); Total_Ratio =
(float)Total_ProbE/Total_ProbNE; Total_AgreeWgt =
LOG10(Total_Ratio)/LOG10(2); Total_CmpRatio = (float)(1-
Total_ProbE)/(1-Total_ProbNE); Total_DisAgWgt =
LOG10(Total_CmpRatio)/LOG10(2); GENERATE Name as Name, NameFreq as
NameFreq, NoNameFreq as NoNameFreq, AgreeE as AgreeE, NamePairs as
NamePairs, AgreeNE as AgreeNE, Total_Pairs as Total_Pairs,
Total_EPairs as Total_EPairs, Total_AgreeE as Total_AgreeE,
Total_NamePairs as Total_NamePairs, Total_AgreeNE as Total_AgreeNE,
Total_NEPairs as Total_NEPairs, ProbE as ProbE, ProbNE as ProbNE,
Total_ProbE as Total_ProbE, Total_ProbNE as Total_ProbNE, Ratio as
Ratio, AgreeWgt as AgreeWgt, CmpRatio as CmpRatio, DisAgWgt as
DisAgWgt, Total_Ratio as Total_Ratio, Total_AgreeWgt as
Total_AgreeWgt, Total_CmpRatio as Total_CmpRatio, Total_DisAgWgt as
Total_DisAgWgt;};
```

```
dump Table_6;
```

```
(NA,3,8,1,3,2,300,37,20,40,20,263,0.027027027681469917,0.0076045626774430275,0.5405405,0.076045625,3.5540541682480464,1.8294656700180507,0.9804287207399688,-
0.02851534766525561,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(Jim,2,8,0,1,1,300,37,20,40,20,263,0.001,0.0038022813387215137,0.5405405,0.076045625,0.2630000145738014,-1.9268652154247259,1.0028129899926799,0.004052589434
311569,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(Mary,4,8,6,6,0,300,37,20,40,20,263,0.1621621549129486,0.001,0.5405405,0.076045625,162.1621549129486,7.341293355260799,0.8386764917765055,-0.2538136772071985
5,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(Harry,1,8,0,0,0,300,37,20,40,20,263,0.001,0.001,0.5405405,0.076045625,1.0000000474974513,6.852433582578467E-8,1.0000000128874909,1.859271902225742E-8,7.1081
08,2.829465610584824,0.49727508,-1.0078839475174377)
(James,7,8,9,21,12,300,37,20,40,20,263,0.2432432472705841,0.045627377927303314,0.5405405,0.076045625,5.33108099391852,2.414428100796611,0.7929363909122012,-0
.33472295682222447,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(Marie,3,8,1,3,2,300,37,20,40,20,263,0.027027027681469917,0.0076045626774430275,0.5405405,0.076045625,3.5540541682480464,1.8294656700180507,0.980428720739968
8,-0.02851534766525561,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(Harold,1,8,0,0,0,300,37,20,40,20,263,0.001,0.001,0.5405405,0.076045625,1.0000000474974513,6.852433582578467E-8,1.0000000128874909,1.859271902225742E-8,7.108
108,2.829465610584824,0.49727508,-1.0078839475174377)
(William,4,8,3,6,3,300,37,20,40,20,263,0.0810810774564743,0.011406844481825829,0.5405405,0.076045625,7.108107556446331,2.8294655116954806,0.929521818468094,-
0.10543936527107774,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
```

7. Calculate Total_AgreeWgt, Total_DisAgWgt

7.1. Group Table_6 all.

```
Total_Weight_group = GROUP Table_6 all;

dump Total_Weight_group;
```

```
(all,{(William,4,8,3,6,3,300,37,20,40,20,263,0.0810810774564743,0.011406844481825829,0.5405405,0.076045625,7.108107556446331,2.8294655116954806,0.92952181846
8094,-0.10543936527107774,7.108108,2.829465610584824,0.49727508,-1.0078839475174377),(Harold,1,8,0,0,0,300,37,20,40,20,263,0.001,0.001,0.5405405,0.076045625,
1.0000000474974513,6.852433582578467E-8,1.0000000128874909,1.859271902225742E-8,7.108108,2.829465610584824,0.49727508,-1.0078839475174377),(Marie,3,8,1,3,2,3
00,37,20,40,20,263,0.027027027681469917,0.0076045626774430275,0.5405405,0.076045625,3.5540541682480464,1.8294656700180507,0.9804287207399688,-0.0285153476652
5561,7.108108,2.829465610584824,0.49727508,-1.0078839475174377),(James,7,8,9,21,12,300,37,20,40,20,263,0.2432432472705841,0.045627377927303314,0.5405405,0.07
6045625,5.33108099391852,2.414428100796611,0.7929363909122012,-0.33472295682222447,7.108108,2.829465610584824,0.49727508,-1.0078839475174377),(Harry,1,8,0,0,
0,300,37,20,40,20,263,0.001,0.001,0.5405405,0.076045625,1.0000000474974513,6.852433582578467E-8,1.0000000128874909,1.859271902225742E-8,7.108108,2.8294656105
84824,0.49727508,-1.0078839475174377),(Mary,4,8,6,6,0,300,37,20,40,20,263,0.1621621549129486,0.001,0.5405405,0.076045625,162.1621549129486,7.341293355260799,
0.8386764917765055,-0.25381367720719855,7.108108,2.829465610584824,0.49727508,-1.0078839475174377),(Jim,2,8,0,1,1,300,37,20,40,20,263,0.001,0.003802281338721
5137,0.5405405,0.076045625,0.2630000145738014,-1.9268652154247259,1.0028129899926799,0.004052589434311569,7.108108,2.829465610584824,0.49727508,-1.0078839475
174377),(NA,3,8,1,3,2,300,37,20,40,20,263,0.027027027681469917,0.0076045626774430275,0.5405405,0.076045625,3.5540541682480464,1.8294656700180507,0.9804287207
399688,-0.02851534766525561,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)})
```

7.2. Use the FOREACH operator generate the relation only have Name, Total_AgreeWgt and Total_DisAgWgt.

Note: Total_AgreeWgt and Total_DisAgWgt are duplicate values in Table_6, you should use MAX function[7][14] to remove duplicates to get unique value.

```
Total_Weight = FOREACH Total_Weight_group GENERATE Table_6.Name as
Name, MAX(Table_6.Total_AgreeWgt) as Total_AgreeWgt,
MAX(Table_6.Total_DisAgWgt) as Total_DisAgWgt;

dump Total_Weight;
```

```
({(William),(Harold),(Marie),(James),(Harry),(Mary),(Jim),(NA)},2.829465610584824,-1.0078839475174377)
```

8. Calculate MissingAgreeWgt, MissingDisAgWgt

8.1. Use SPLIT operator[7] to partitions Table_6 into two relations, one just contains missing values-NA (MissingValue_Select), the other includes the remaining values (MissingValue_No).

```
SPLIT Table_6 into MissingValue_Select if Name == 'NA',
MissingValue_No if Name != 'NA';

dump MissingValue_Select;
```

```
(NA,3,8,1,3,2,300,37,20,40,20,263,0.027027027681469917,0.0076045626774430275,0.5405405,0.076045625,3.5540541682480464,1.8294656700180507,0.9804287207399688,-
0.02851534766525561,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
```

```
dump MissingValue_No;
```

```
(Jim,2,8,0,1,1,300,37,20,40,20,263,0.001,0.0038022813387215137,0.5405405,0.076045625,0.2630000145738014,-1.9268652154247259,1.0028129899926799,0.004052589434311569,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(Mary,4,8,6,6,0,300,37,20,40,20,263,0.1621621549129486,0.001,0.5405405,0.076045625,162.1621549129486,7.341293355260799,0.8386764917765055,-0.25381367720719855,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(Harry,1,8,0,0,0,300,37,20,40,20,263,0.001,0.001,0.5405405,0.076045625,1.0000000474974513,6.852433582578467E-8,1.0000000128874909,1.859271902225742E-8,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(James,7,8,9,21,12,300,37,20,40,20,263,0.2432432472705841,0.045627377927303314,0.5405405,0.076045625,5.33108099391852,2.414428100796611,0.7929363909122012,-0.33472295682222447,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(Marie,3,8,1,3,2,300,37,20,40,20,263,0.027027027681469917,0.0076045626774430275,0.5405405,0.076045625,3.5540541682480464,1.8294656700180507,0.9804287207399688,-0.02851534766525561,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(Harold,1,8,0,0,0,300,37,20,40,20,263,0.001,0.001,0.5405405,0.076045625,1.0000000474974513,6.852433582578467E-8,1.0000000128874909,1.859271902225742E-8,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(William,4,8,3,6,3,300,37,20,40,20,263,0.0810810774564743,0.011406844481825829,0.5405405,0.076045625,7.108107556446331,2.8294655116954806,0.929521818468094,-0.10543936527107774,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
```

8.2. Use the FOREACH operator on MissingValue_Select generate the relation only have the missing values Name, MissingAgreeWgt and MissingDisAgWgt.

```
MissingValue_Weight = FOREACH MissingValue_Select GENERATE Name as Name, AgreeWgt as MissingAgreeWgt, DisAgWgt as MissingDisAgWgt;

dump MissingValue_Weight;
```

```
(NA,1.8294656700180507,-0.02851534766525561)
```

9. Calculate AgreeWgt, DisAgWgt, Total_RemnAgreeWgt, Total_RemnDisAgWgt

9.1. Sort the MissingValue_No in descending ORDER by[7] NameFreq and in ascending ORDER by Name at the same time.

```
Table_7 = ORDER MissingValue_No by NameFreq DESC, Name;

dump Table_7;
```

```
(James,7,8,9,21,12,300,37,20,40,20,263,0.2432432472705841,0.045627377927303314,0.5405405,0.076045625,5.33108099391852,2.414428100796611,0.7929363909122012,-0.33472295682222447,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(Mary,4,8,6,6,0,300,37,20,40,20,263,0.1621621549129486,0.001,0.5405405,0.076045625,162.1621549129486,7.341293355260799,0.8386764917765055,-0.25381367720719855,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(William,4,8,3,6,3,300,37,20,40,20,263,0.0810810774564743,0.011406844481825829,0.5405405,0.076045625,7.108107556446331,2.8294655116954806,0.929521818468094,-0.10543936527107774,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(Marie,3,8,1,3,2,300,37,20,40,20,263,0.027027027681469917,0.0076045626774430275,0.5405405,0.076045625,3.5540541682480464,1.8294656700180507,0.9804287207399688,-0.02851534766525561,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(Jim,2,8,0,1,1,300,37,20,40,20,263,0.001,0.0038022813387215137,0.5405405,0.076045625,0.2630000145738014,-1.9268652154247259,1.0028129899926799,0.004052589434311569,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(Harold,1,8,0,0,0,300,37,20,40,20,263,0.001,0.001,0.5405405,0.076045625,1.0000000474974513,6.852433582578467E-8,1.0000000128874909,1.859271902225742E-8,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(Harry,1,8,0,0,0,300,37,20,40,20,263,0.001,0.001,0.5405405,0.076045625,1.0000000474974513,6.852433582578467E-8,1.0000000128874909,1.859271902225742E-8,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
```

9.2. Select the top 60 percent frequency of Name values by using LIMIT operator [7][15]. To change the frequency percentage, simply replace 0.6 to any other number between 0 and 1. Since the top percentage frequency could be a number with digits, I use ROUND function [7] which returns the value of an expression rounded to an integer.

When calculate top frequency value weights, the missing value is not included in the list. Therefore, in this example, the number of names excluding missing values equals to the total number of name frequency - 1.

```
Top = LIMIT Table_7 ROUND(0.6 * (Total_NamePairs_Count.NoNameFreq - 1));

dump Top;
```

```
(James,7,8,9,21,12,300,37,20,40,20,263,0.2432432472705841,0.045627377927303314,0.5405405,0.076045625,5.33108099391852,2.414428100796611,0.7929363909122012,-0.33472295682222447,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(Mary,4,8,6,6,0,300,37,20,40,20,263,0.1621621549129486,0.001,0.5405405,0.076045625,162.1621549129486,7.341293355260799,0.8386764917765055,-0.25381367720719855,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(William,4,8,3,6,3,300,37,20,40,20,263,0.0810810774564743,0.011406844481825829,0.5405405,0.076045625,7.108107556446331,2.8294655116954806,0.929521818468094,-0.10543936527107774,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(Marie,3,8,1,3,2,300,37,20,40,20,263,0.027027027681469917,0.0076045626774430275,0.5405405,0.076045625,3.5540541682480464,1.8294656700180507,0.9804287207399688,-0.02851534766525561,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
```

9.3. Use the FOREACH operator on Top generate the relation only have the Name, AgreeWgt and DisAgWgt. This is the agreement weight (AgreeWgt) and disagreement weight (DisAgWgt) for the top 60 percent frequency of Name values.

```
Top_Weight = FOREACH Top GENERATE Name as Name, AgreeWgt as
AgreeWgt, DisAgWgt as DisAgWgt;
```

```
dump Top_Weight;
```

```
(James,2.414428100796611,-0.33472295682222447)
(Mary,7.341293355260799,-0.25381367720719855)
(William,2.8294655116954806,-0.10543936527107774)
(Marie,1.8294656700180507,-0.02851534766525561)
```

9.4. Sort the MissingValue_No in ascending ORDER by NameFreq and in descending ORDER by Name at the same time.

```
Table_8 = ORDER MissingValue_No by NameFreq, Name DESC;
```

```
dump Table_8;
```

```
(Harry,1,8,0,0,0,300,37,20,40,20,263,0.001,0.001,0.5405405,0.076045625,1.0000000474974513,6.852433582578467E-8,1.0000000128874909,1.859271902225742E-8,7.1081
08,2.829465610584824,0.49727508,-1.0078839475174377)
(Harold,1,8,0,0,0,300,37,20,40,20,263,0.001,0.001,0.5405405,0.076045625,1.0000000474974513,6.852433582578467E-8,1.0000000128874909,1.859271902225742E-8,7.108
108,2.829465610584824,0.49727508,-1.0078839475174377)
(Jim,2,8,0,1,1,300,37,20,40,20,263,0.001,0.0038022813387215137,0.5405405,0.076045625,0.2630000145738014,-1.9268652154247259,1.0028129899926799,0.00405258943
4311569,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(Marie,3,8,1,3,2,300,37,20,40,20,263,0.027027027681469917,0.0076045626774430275,0.5405405,0.076045625,3.554054168248046 4,1.8294656700180507,0.980428720739968
8,-0.02851534766525561,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(William,4,8,3,6,3,300,37,20,40,20,263,0.0810810774564743,0.011406844481825829,0.5405405,0.076045625,7.108107556446331,2.8294655116954806,0.929521818468094,-
0.10543936527107774,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(Mary,4,8,6,6,0,300,37,20,40,20,263,0.1621621549129486,0.001,0.5405405,0.076045625,162.1621549129486,7.341293355260799,0.8386764917765055,-0.2538136772071985
5,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
(James,7,8,9,21,12,300,37,20,40,20,263,0.2432432472705841,0.045627377927303314,0.5405405,0.076045625,5.33108099391852,2.414428100796611,0.7929363909122012,-0
.33472295682222447,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
```

9.5. Select the remaining 40 percent of Name values by using LIMIT operator.

Note: Table_8 is also excludes missing values, so the number of names excluding missing values equal to the total number of name frequency - 1.

```
Remaining_a = LIMIT Table_8 ((Total_NamePairs_Count.NoNameFreq - 1)
- ROUND(0.6 * (Total_NamePairs_Count.NoNameFreq - 1)));
```

```
dump Remaining_a;
```

```
(Harry,1,8,0,0,0,300,37,20,40,20,263,0.001,0.001,0.5405405,0.076045625,1.0000000474974513,6.852433582578467E-8,1.0000000128874909,1.859271902225742E-8,7.1081
08,2.829465610584824,0.49727508,-1.0078839475174377)
(Harold,1,8,0,0,0,300,37,20,40,20,263,0.001,0.001,0.5405405,0.076045625,1.0000000474974513,6.852433582578467E-8,1.0000000128874909,1.859271902225742E-8,7.108
108,2.829465610584824,0.49727508,-1.0078839475174377)
(Jim,2,8,0,1,1,300,37,20,40,20,263,0.001,0.0038022813387215137,0.5405405,0.076045625,0.2630000145738014,-1.9268652154247259,1.0028129899926799,0.00405258943
4311569,7.108108,2.829465610584824,0.49727508,-1.0078839475174377)
```

9.6. Group Remaining_a all.

```
Remaining_b = GROUP Remaining_a all;
```

```
dump Remaining_b;
```

```
(all,{(Jim,2,8,0,1,1,300,37,20,40,20,263,0.001,0.0038022813387215137,0.5405405,0.076045625,0.2630000145738014,-1.9268652154247259,1.0028129899926799,0.004052
589434311569,7.108108,2.829465610584824,0.49727508,-1.0078839475174377),(Harold,1,8,0,0,0,300,37,20,40,20,263,0.001,0.001,0.5405405,0.076045625,1.00000004749
74513,6.852433582578467E-8,1.0000000128874909,1.859271902225742E-8,7.108108,2.829465610584824,0.49727508,-1.0078839475174377),(Harry,1,8,0,0,0,300,37,20,40,2
0,263,0.001,0.001,0.5405405,0.076045625,1.0000000474974513,6.852433582578467E-8,1.0000000128874909,1.859271902225742E-8,7.108108,2.829465610584824,0.49727508
,-1.0078839475174377)})
```

9.7. Use the FOREACH operator, Total_RemnAgreeE, Total_RemnAgreeNE, Total_NEPairs, Total_RemnProbE and Total_RemnProbNE are performed within the nested block, and then generate Name, Total_RemnProbE, Total_RemnProbNE, Total_RemnDisAgWgt.

Note: when divide two integers to get a float number by using FLOAT to convert the integers to float first and then calculate; when calculate the probability using 0.001 instead of zero, by using bincond operator; our research group use the Total_DisAgWgt as the Total_RemnDisAgWgt, and since the Total_DisAgWgt are duplicate values, you should use MAX function to remove duplicates to get unique value.

```
Total_Remaining_Weight_a = FOREACH Remaining_b {Total_RemnAgreeE =
SUM(Remaining_a.AgreeE); Total_RemnAgreeNE =
SUM(Remaining_a.AgreeNE); Total_NEPairs =
Total_Pairs_Count.Total_Pairs - Total_EPairs_Count.Total_EPairs;
Total_RemnProbE =
(float)Total_RemnAgreeE/Total_EPairs_Count.Total_EPairs;
Total_RemnProbNE = (float)Total_RemnAgreeNE/Total_NEPairs; GENERATE
Remaining_a.Name as Name, (Total_RemnProbE == 0.0 ?
0.001:Total_RemnProbE) as Total_RemnProbE, (Total_RemnProbNE ==
0.0 ? 0.001:Total_RemnProbNE) as Total_RemnProbNE,
MAX(Remaining_a.Total_DisAgWgt) as Total_RemnDisAgWgt;};
```

```
dump Total_Remaining_Weight_a;
```

```
({(Jim),(Harold),(Harry)},0.001,0.0038022813387215137,-1.0078839475174377)
```

9.8. Use the FOREACH operator, Total_RemnAgreeWgt is performed within the nested block, and then generate Name, Total_RemnAgreeWgt and Total_RemnDisAgWgt.

```
Total_Remaining_Weight = FOREACH Total_Remaining_Weight_a
{Total_RemnAgreeWgt =
(float)LOG10(Total_RemnProbE/Total_RemnProbNE)/LOG10(2); GENERATE
Name as Name, Total_RemnAgreeWgt as Total_RemnAgreeWgt,
Total_RemnDisAgWgt as Total_RemnDisAgWgt;};
```

```
dump Total_Remaining_Weight;
```

```
({(Jim),(Harold),(Harry)},-1.9268653553356543,-1.0078839475174377)
```

Step 5: Output the Results to HDFS

In the example above, I use DUMP statement to display results to the terminal screen. However, if the data set is large, the output will not able to display full in the terminal screen. In this case, we can use the STORE operator and the store functions to save results to the file system (Hue).

- Syntax: STORE alias INTO 'directory' [USING function].

For example, the code in step 4-7 is changed to:

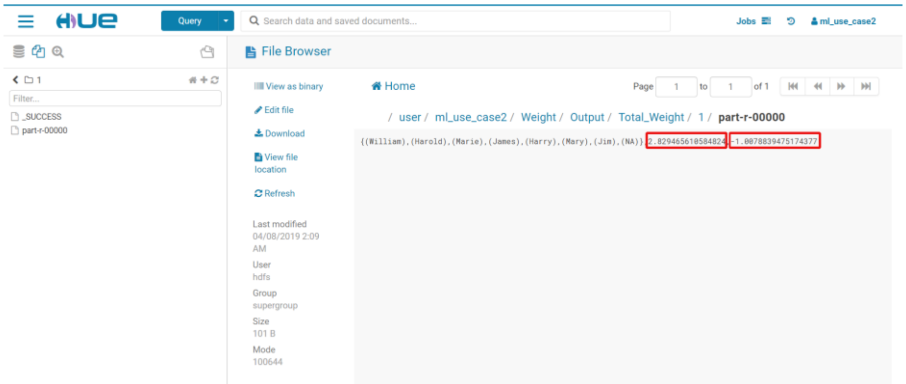
```
Total_Weight_group = GROUP Table_6 all;
```

```
dump Total_Weight_group;
```

```
Total_Weight = FOREACH Total_Weight_group GENERATE Table_6.Name
as Name, MAX(Table_6.Total_AgreeWgt) as Total_AgreeWgt,
MAX(Table_6.Total_DisAgWgt) as Total_DisAgWgt;
```

```
STORE Total_Weight INTO
'/user/ml_use_case2/Weight/Output/Total_Weight/1' using
PigStorage (',');
```

Then the result will be saved in Hue.



References

[1] Wang, P., Pullen, D., Talburt, J.R., Wu, N.(2014). *Iterative Approach to Weight Calculation in Probabilistic Entity Resolution.* Proceedings of the 19th International Conference on Information Quality (ICIQ-2014).

[2] Alsarkhi, A. & Talburt, J.R.(2018). *A Method for Implementing Probabilistic Entity Resolution*. (IJACSA) International Journal of Advanced Computer Science and Applications.

[3] I. P. Fellegi and A. B. Sunter. *A Theory for Record Linkage*. Journal of the American Statistical Association, vol. 64, pp. 1183--1210, 1969.

[4] Kobayashi, Fumiko & Eram, Aziz & Talburt, John.(2018). *Entity Resolution Using Logistic Regression as an extension to the Rule-Based Oyster System*. 2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), 146-151. 10.1109/MIPR.2018.00033.

[5] Talburt, J.R.(2018). *Guide to Scoring Rule in OYSTER (v6)* https://bitbucket.org/oysterer/oyster/downloads/.

[6] Chris(2017). *How does Apache Pig handle nulls*. https://medium.com/codelog/how-does-pig-handle-the-null-6bf78ec655ad .

[7] *Pig Latin Basics - Apache Pig! - The Apache Software Foundation!* https://pig.apache.org/docs/r0.17.0/basic.html .

[8] *How can I add a header row to files created from Pig (Hadoop)?* https://stackoverflow.com/questions/14204275/how-can-i-add-a-header-row-to-files-created-from-pig-hadoop .

[9] Jarodhu(2004). *W3Cschool- Apache Pig Tutorial(教程)*. https://www.w3cschool.cn/apache_pig/? .

[10]*Apache Pig 101*. https://courses.cognitiveclass.ai/courses/course-v1:BigDataUniversity+BD0121EN+2016/info .

[11]*Apache Pig 的一些基础概念及用法总结（1）* https://www.codelast.com/%E5%8E%9F%E5%88%9Bpig%E4%B8%AD%E7%9A%84%E4%B8%80%E4%BA%9B%E5%9F%BA%E7%A1%80%E6%A6%82%E5%BF%B5%E5%8F%8A%E7%94%A8%E6%B3%95%E6%80%BB%E7%BB%93/ .

[12]group-by - 在 Apache Pig 中，计数/求和. https://ask.helplib.com/group-by/post_10262484 .

[13]*Index and logarithm(指数与对数)*. https://www.shuxuele.com/algebra/exponents-logarithms.html .
[14]*get MAX from SUM in PIG.* https://stackoverflow.com/questions/32859534/get-max-from-sum-in-pig.
[15]*Limit but not Order in PIG.* https://stackoverflow.com/questions/27053924/limit-but-not-order-in-pig .