Ujjwal Pandit, Hari Prasath KP

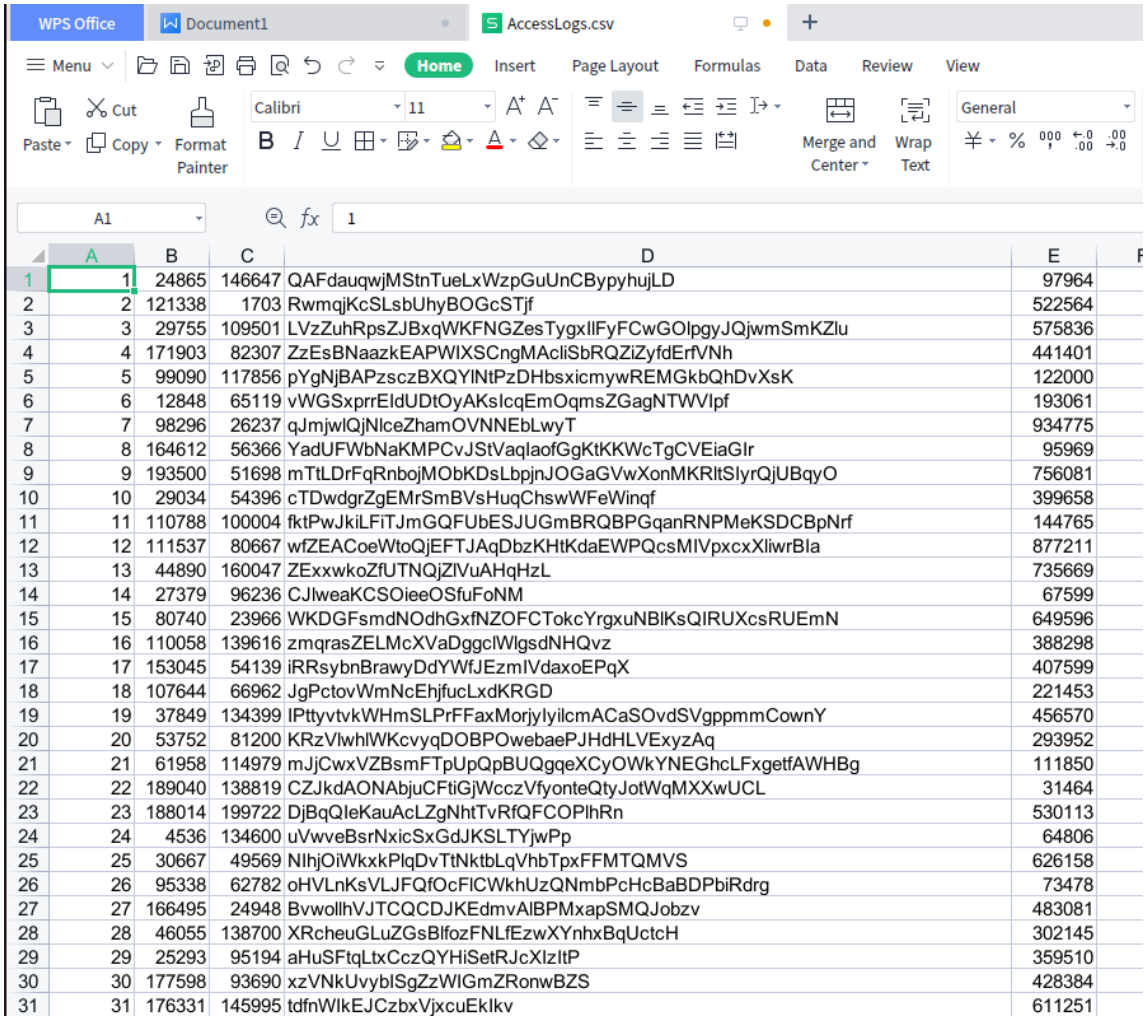
Project 1

CS 585: Big Data Management

The LinkBook Network

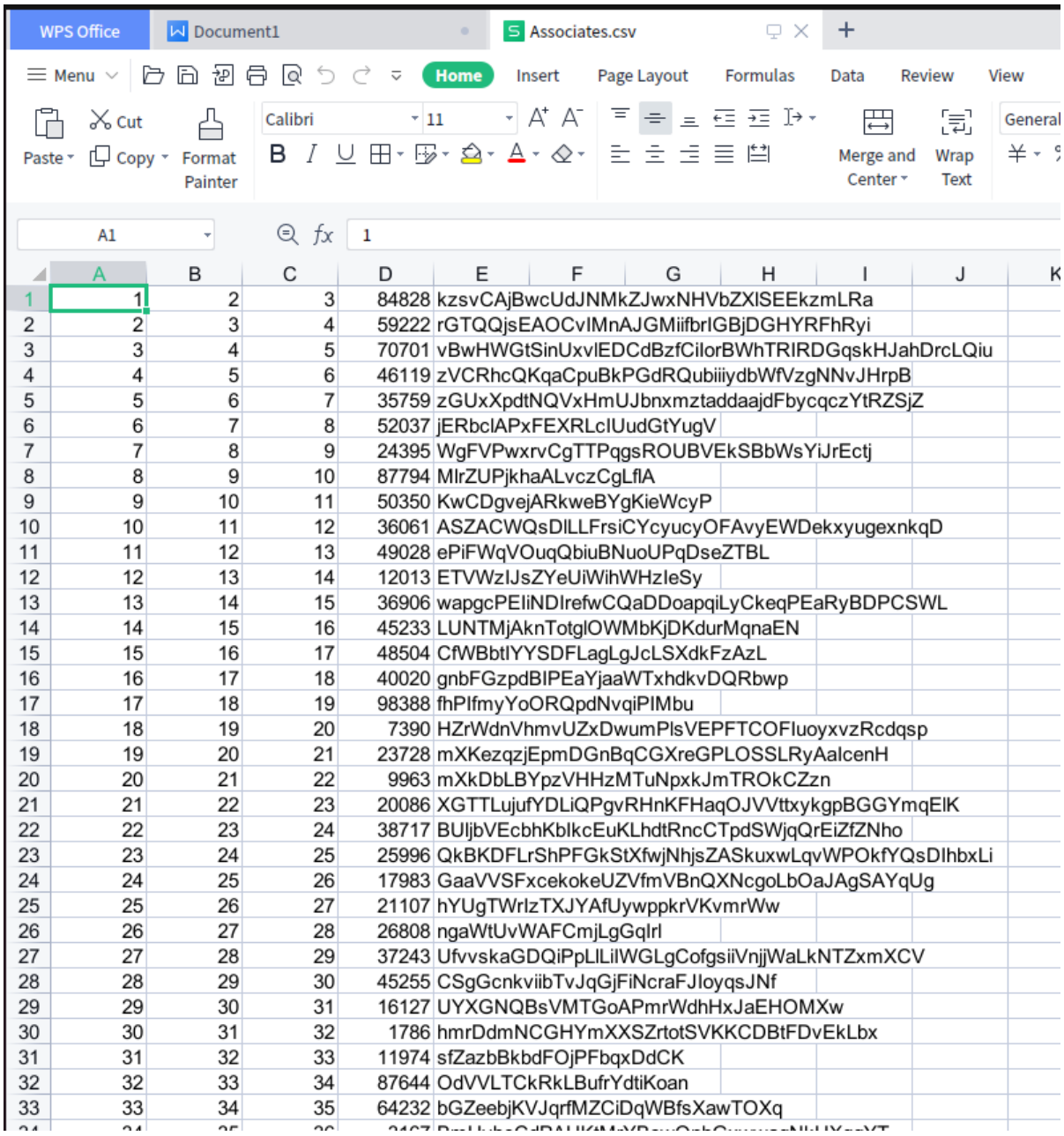
# Project Documentation

## 1- Creating Datasets for a LinkBook Big Data Application [15 Points]



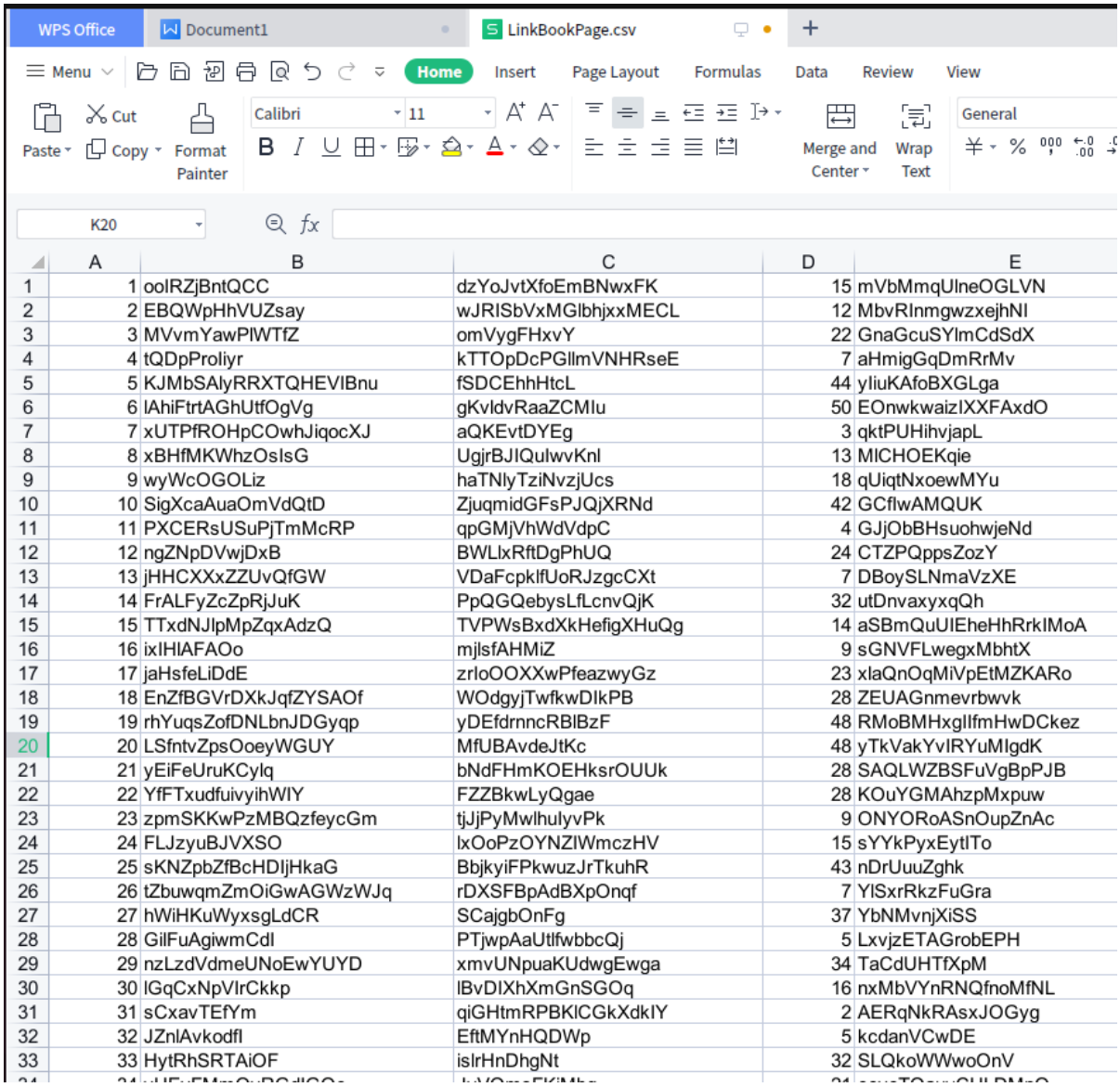

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | 1 | 24865 | 146647 | QAFdauqwjMStnTueLxWzpGuUnCBypyhujLD | 97964 |
| 2 | 2 | 121338 | 1703 | RwmqjKcSLsbUhyBOGcSTjf | 522564 |
| 3 | 3 | 29755 | 109501 | LVzZuhRpsZJBxqWKFNGZesTygxIlFyFCwGOlpgyJQjwmSmKZlu | 575836 |
| 4 | 4 | 171903 | 82307 | ZzEsBNaazkEAPWIXSCngMAcliSbRQZiZyfdErfVNh | 441401 |
| 5 | 5 | 99090 | 117856 | pYgNjBAPzsczBXQYlNtPzDHbsxicmywREMGkbQhDvXsK | 122000 |
| 6 | 6 | 12848 | 65119 | vWGSxprrEldUDtOyAKsIcqEmOqmsZGagNTWVlpf | 193061 |
| 7 | 7 | 98296 | 26237 | qJmjwlQjNlceZhamOVNNEbLwyT | 934775 |
| 8 | 8 | 164612 | 56366 | YadUFWbNaKMPCvJStVaqlaofGgKtKKWcTgCVEiaGlr | 95969 |
| 9 | 9 | 193500 | 51698 | mTtLDrFqRnbojMObKDsLbpjnJOGaGVwXonMKRltSlyrQjUBqyO | 756081 |
| 10 | 10 | 29034 | 54396 | cTDwdgrZgEMrSmBVsHuqChswWFeWinqf | 399658 |
| 11 | 11 | 110788 | 100004 | fktPwJkiLFiTJmGQFUbESJUGmBRQBPGqanRNPMeKSDCBpNrf | 144765 |
| 12 | 12 | 111537 | 80667 | wfZEACoeWtoQjEFTJAqDbzKHtKdaEWPQcsMIVpxcxXliwrBla | 877211 |
| 13 | 13 | 44890 | 160047 | ZExxwkoZfUTNQjZlVuAHqHzL | 735669 |
| 14 | 14 | 27379 | 96236 | CJlweaKCSOieeOSfuFoNM | 67599 |
| 15 | 15 | 80740 | 23966 | WKDGFsmdNOdhGxfNZOFCTokcYrgxuNBlKsQIRUXcsRUEmN | 649596 |
| 16 | 16 | 110058 | 139616 | zmqrasZELMcXVaDggclWlgsdNHQvz | 388298 |
| 17 | 17 | 153045 | 54139 | iRRsybnBrawyDdYWfJEzmIVdaxoEPqX | 407599 |
| 18 | 18 | 107644 | 66962 | JgPctovWmNcEhjfucLxdKRGD | 221453 |
| 19 | 19 | 37849 | 134399 | lPttyvtvkWHmSLPrFFaxMorjylyilcmACaSOvdSVgppmmCownY | 456570 |
| 20 | 20 | 53752 | 81200 | KRzVlwhlWKcvyqDOBPOwebaePJHdHLVExyzAq | 293952 |
| 21 | 21 | 61958 | 114979 | mJjCwxVZBsmFTpUpQpBUQgqeXCyOWkYNEGhcLFxgetfAWHBg | 111850 |
| 22 | 22 | 189040 | 138819 | CZJkdAONAbjuCFtiGjWcczVfyonteQtyJotWqMXXwUCL | 31464 |
| 23 | 23 | 188014 | 199722 | DjBqQIeKauAcLZgNhtTvRfQFCOPlhRn | 530113 |
| 24 | 24 | 4536 | 134600 | uVwveBsrNxicSxGdJKSLTYjwPp | 64806 |
| 25 | 25 | 30667 | 49569 | NIhjOiWkxkPlqDvTtNktbLqVhbTpxFFMTQMVS | 626158 |
| 26 | 26 | 95338 | 62782 | oHVLnKsVLJFQfOcFlCWkhUzQNmbPcHcBaBDPbiRdrg | 73478 |
| 27 | 27 | 166495 | 24948 | BvwollhVJTCQCDJKEdmvAlBPMxapSMQJobzv | 483081 |
| 28 | 28 | 46055 | 138700 | XRcheuGLuZGsBlfozFNLfEzwXYnhxBqUctcH | 302145 |
| 29 | 29 | 25293 | 95194 | aHuSFtqLtxCczQYHiSetRJcXlzItP | 359510 |
| 30 | 30 | 177598 | 93690 | xzVNkUvyblSgZzWlGmZRonwBZS | 428384 |
| 31 | 31 | 176331 | 145995 | tdfnWlkEJCzbxVjxcuEklkv | 611251 |

## AccessLogs.csv



| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 84828 | kzsvCAjBwcUdJNMkZJwxNHVbZXlSEEkzmLRa |
| 2 | 2 | 3 | 4 | 59222 | rGTQQjsEAOCvIMnAJGMiifbrIGBjDGHYRFhRyi |
| 3 | 3 | 4 | 5 | 70701 | vBwHWGtSinUxvlEDCdBzfCilorBWhTRIRDGqskHJahDrcLQiu |
| 4 | 4 | 5 | 6 | 46119 | zVCRhcQKqaCpuBkPGdRQubiiiydbWfVzgNNvJHrpB |
| 5 | 5 | 6 | 7 | 35759 | zGUxXpdtNQVxHmUJbnxmztaddaajdFbycqczYtRZSjZ |
| 6 | 6 | 7 | 8 | 52037 | jERbclAPxFEXRLclUudGtYugV |
| 7 | 7 | 8 | 9 | 24395 | WgFVPwxrvCgTTPqgsROUBVEkSBbWsYiJrEctj |
| 8 | 8 | 9 | 10 | 87794 | MlrZUPjkhaALvczCgLflA |
| 9 | 9 | 10 | 11 | 50350 | KwCDgvejARkweBYgKieWcyP |
| 10 | 10 | 11 | 12 | 36061 | ASZACWQsDlLLFrsiCYcyucyOFAvyEWDekxyugexnkqD |
| 11 | 11 | 12 | 13 | 49028 | ePiFWqVOuqQbiuBNuoUPqDseZTBL |
| 12 | 12 | 13 | 14 | 12013 | ETVWzIJsZYeUiWihWHzIeSy |
| 13 | 13 | 14 | 15 | 36906 | wapgcPEIiNDIrefwCQaDDoapqiLyCkeqPEaRyBDPCSWL |
| 14 | 14 | 15 | 16 | 45233 | LUNTMjAknTotglOWMbKjDKdurMqnaEN |
| 15 | 15 | 16 | 17 | 48504 | CfWBbtlYYSDFLagLgJcLSXdkFzAzL |
| 16 | 16 | 17 | 18 | 40020 | gnbFGzpdBIPEaYjaaWTxhdkvDQRbwp |
| 17 | 17 | 18 | 19 | 98388 | fhPIfmyYoORQpdNvqiPIMbu |
| 18 | 18 | 19 | 20 | 7390 | HZrWdnVhmvUZxDwumPlsVEPFTCOFluoyxvzRcdqsp |
| 19 | 19 | 20 | 21 | 23728 | mXKezqzjEpmDGnBqCGXreGPLOSSLRyAaIcenH |
| 20 | 20 | 21 | 22 | 9963 | mXkDbLBYpzVHHzMTuNpxkJmTROkCZzn |
| 21 | 21 | 22 | 23 | 20086 | XGTTLujufYDLiQPgvRHnKFHaqOJVVttxykgpBGGYmqElK |
| 22 | 22 | 23 | 24 | 38717 | BUljbVEcbhKblkcEuKLhdtRncCTpdSWjqQrEiZfZNho |
| 23 | 23 | 24 | 25 | 25996 | QkBKDFLrShPFGkStXfwjNhjsZASkuxwLqvWPOkfYQsDIhbxLi |
| 24 | 24 | 25 | 26 | 17983 | GaaVVSFxcekokeUZVfmVBnQXNcgoLbOaJAgSAYqUg |
| 25 | 25 | 26 | 27 | 21107 | hYUgTWrlzTXJYAfUywppkrVKvmrWw |
| 26 | 26 | 27 | 28 | 26808 | ngaWtUvWAFCmjLgGqlrl |
| 27 | 27 | 28 | 29 | 37243 | UfvvskaGDQiPpLlLilWGLgCofgsiiVnjjWaLkNTZxmXCV |
| 28 | 28 | 29 | 30 | 45255 | CSgGcnkviibTvJqGjFiNcraFJloyqsJNf |
| 29 | 29 | 30 | 31 | 16127 | UYXGNQBsVMTGoAPmrWdhHxJaEHOMXw |
| 30 | 30 | 31 | 32 | 1786 | hmrDdmNCGHYmXXSZrtotSVKKCDBtFDvEkLbx |
| 31 | 31 | 32 | 33 | 11974 | sfZazbBkbdFOjPFbqxDdCK |
| 32 | 32 | 33 | 34 | 87644 | OdVVLTCkRkLBufrYdtiKoan |
| 33 | 33 | 34 | 35 | 64232 | bGZeebjKVJqrfMZCiDqWBfsXawTOXq |

## Associates.csv




| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | 1 | oolRZjBntQCC | dzYoJvtXfoEmBNwxFK | 15 | mVbMmqUlneOGLVN |
| 2 | 2 | EBQWpHhVUZsay | wJRISbVxMGlbhjxxMECL | 12 | MbvRInmgwzxejhNI |
| 3 | 3 | MVvmYawPlWTfZ | omVygFHxvY | 22 | GnaGcuSYlmCdSdX |
| 4 | 4 | tQDpProliyr | kTTOpDcPGllmVNHRseE | 7 | aHmigGqDmRrMv |
| 5 | 5 | KJMbSAlyRRXTQHEVlBnu | fSDCEhhHtcL | 44 | yliuKAfoBXGLga |
| 6 | 6 | lAhiFtrtAGhUtfOgVg | gKvldvRaaZCMlu | 50 | EOnwkwaizIXXFAxdO |
| 7 | 7 | xUTPfROHpCOwhJiqocXJ | aQKEvtDYEg | 3 | qktPUHihvjapL |
| 8 | 8 | xBHfMKWhzOsIsG | UgjrBJIQulwvKnl | 13 | MlCHOEKqie |
| 9 | 9 | wyWcOGOLiz | haTNlyTziNvzjUcs | 18 | qUiqtNxoewMYu |
| 10 | 10 | SigXcaAuaOmVdQtD | ZjuqmidGFsPJQjXRNd | 42 | GCflwAMQUK |
| 11 | 11 | PXCERsUSuPjTmMcRP | qpGMjVhWdVdpC | 4 | GJjObBHsuohwjeNd |
| 12 | 12 | ngZNpDVwjDxB | BWLlxRftDgPhUQ | 24 | CTZPQppsZozY |
| 13 | 13 | jHHCXXxZZUvQfGW | VDaFcpklfUoRJzgcCXt | 7 | DBoySLNmaVzXE |
| 14 | 14 | FrALFyZcZpRjJuK | PpQGQebysLfLcnvQjK | 32 | utDnvaxyxqQh |
| 15 | 15 | TTxdNJlpMpZqxAdzQ | TVPWsBxdXkHefigXHuQg | 14 | aSBmQuUlEheHhRrkIMoA |
| 16 | 16 | ixIHlAFAOo | mjlsfAHMiZ | 9 | sGNVFLwegxMbhtX |
| 17 | 17 | jaHsfeLiDdE | zrloOOXXwPfeazwyGz | 23 | xlaQnOqMiVpEtMZKARo |
| 18 | 18 | EnZfBGVrDXkJqfZYSAOf | WOdgyjTwfkwDlkPB | 28 | ZEUAGnmevrbwvk |
| 19 | 19 | rhYuqsZofDNLbnJDGyqp | yDEfdrnncRBlBzF | 48 | RMoBMHxglIfmHwDCkez |
| 20 | 20 | LSfntvZpsOoeyWGUY | MfUBAvdeJtKc | 48 | yTkVakYvIRYuMlgdK |
| 21 | 21 | yEiFeUruKCylq | bNdFHmKOEHksrOUUk | 28 | SAQLWZBSFuVgBpPJB |
| 22 | 22 | YfFTxudfuivyihWlY | FZZBkwLyQgae | 28 | KOuYGMAhzpMxpuw |
| 23 | 23 | zpmSKKwPzMBQzfeycGm | tjJjPyMwlhulyvPk | 9 | ONYORoASnOupZnAc |
| 24 | 24 | FLJzyuBJVXSO | lxOoPzOYNZlWmczHV | 15 | sYYkPyxEytlTo |
| 25 | 25 | sKNZpbZfBcHDljHkaG | BbjkyiFPkwuzJrTkuhR | 43 | nDrUuuZghk |
| 26 | 26 | tZbuwqmZmOiGwAGWzWJq | rDXSFBpAdBXpOnqf | 7 | YlSxrRkzFuGra |
| 27 | 27 | hWiHKuWyxsgLdCR | SCajgbOnFg | 37 | YbNMvnjXiSS |
| 28 | 28 | GilFuAgiwmCdl | PTjwpAaUtlfwbbcQj | 5 | LxvjzETAGrobEPH |
| 29 | 29 | nzLzdVdmeUNoEwYUYD | xmvUNpuaKUdwgEwga | 34 | TaCdUHTfXpM |
| 30 | 30 | lGqCxNpVlrCkkp | lBvDIXhXmGnSGOq | 16 | nxMbVYnRNQfnoMfNL |
| 31 | 31 | sCxavTEfYm | qiGHtmRPBKlCGkXdkIY | 2 | AERqNkRAsxJOGyg |
| 32 | 32 | JZnlAvkodfl | EftMYnHQDWp | 5 | kcdanVCwDE |
| 33 | 33 | HytRhSRTAiOF | islrHnDhgNt | 32 | SLQkoWWwoOnV |

**LinkBookPage.csv**

# 2. Loading Datasets into Hadoop [5 Points]

We had to create datasets and upload them to Hadoop HDFS. We created two directories:

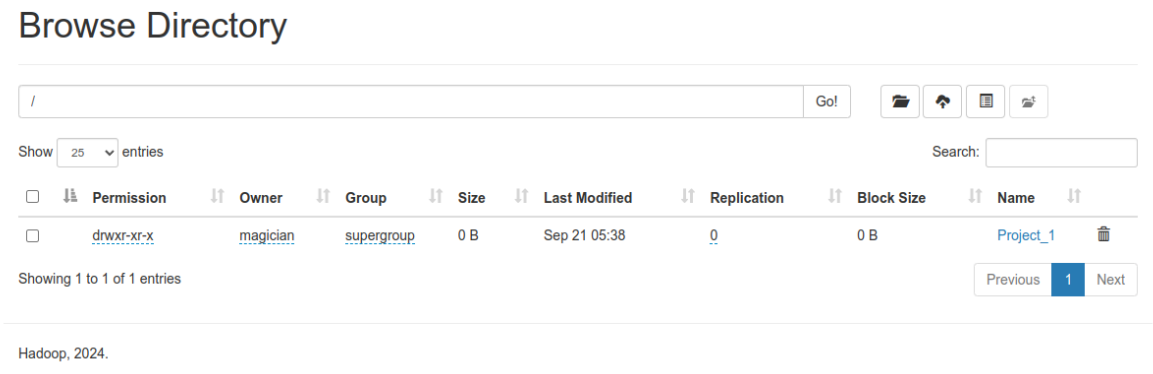
**Project_1/Input_Files :** For storing three input files.As the datasets are huge, we will be using sample dataset to check our work.

**Project_1/Output_Files :** For storing all the intermediate and final output.

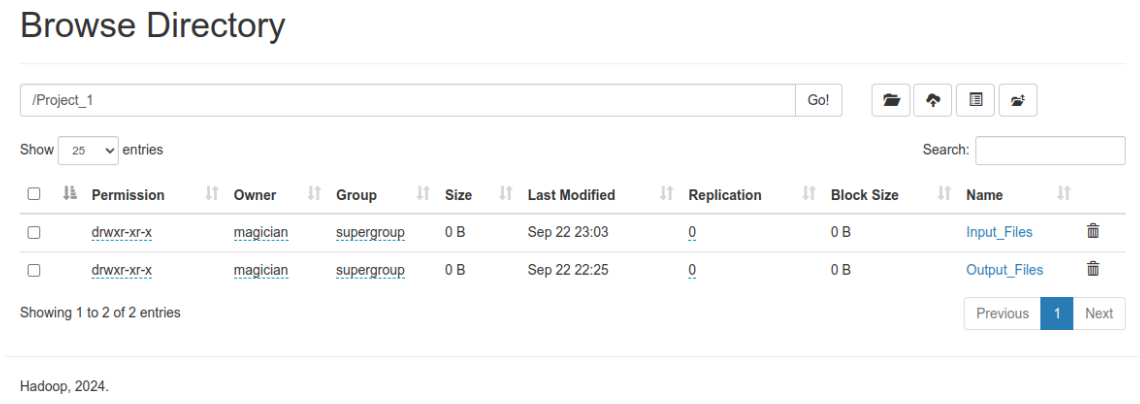
```
magician@eNepal:~/Downloads$ cd DataFiles/
magician@eNepal:~/Downloads/DataFiles$ hadoop fs -put AccessLogs.csv /Project_1/Input_Files/
magician@eNepal:~/Downloads/DataFiles$ hadoop fs -put LinkBookPage.csv /Project_1/Input_Files/
magician@eNepal:~/Downloads/DataFiles$ hadoop fs -put Associates.csv /Project_1/Input_Files/
magician@eNepal:~/Downloads/DataFiles$ hadoop fs -ls
Found 1 items
drwxr-xr-x   - magician supergroup          0 2024-09-22 17:57 Project_1
magician@eNepal:~/Downloads/DataFiles$ hadoop fs -ls /Project_1/
Found 2 items
drwxr-xr-x   - magician supergroup          0 2024-09-22 23:03 /Project_1/Input_Files
drwxr-xr-x   - magician supergroup          0 2024-09-22 22:25 /Project_1/Output_Files
magician@eNepal:~/Downloads/DataFiles$ hadoop fs -ls /Project_1/Input_Files
Found 8 items
-rw-r--r--   1 magician supergroup  646665968 2024-09-22 23:02 /Project_1/Input_Files/AccessLogs.csv
-rw-r--r--   1 magician supergroup 1284483605 2024-09-22 23:03 /Project_1/Input_Files/Associates.csv
-rw-r--r--   1 magician supergroup   11648336 2024-09-22 23:03 /Project_1/Input_Files/LinkBookPage.csv
-rw-r--r--   1 magician supergroup        629 2024-09-22 01:40 /Project_1/Input_Files/New_AccessLogs.csv
-rw-r--r--   1 magician supergroup        647 2024-09-21 05:39 /Project_1/Input_Files/samp_AccessLogs.csv
-rw-r--r--   1 magician supergroup        718 2024-09-21 16:26 /Project_1/Input_Files/samp_AccessLogs1.csv
-rw-r--r--   1 magician supergroup        564 2024-09-21 05:39 /Project_1/Input_Files/samp_Associates.csv
-rw-r--r--   1 magician supergroup        643 2024-09-21 05:40 /Project_1/Input_Files/samp_LinkBookPage.csv
magician@eNepal:~/Downloads/DataFiles$
```

Terminal View of Project 1 Files in Hadoop Browser Overview

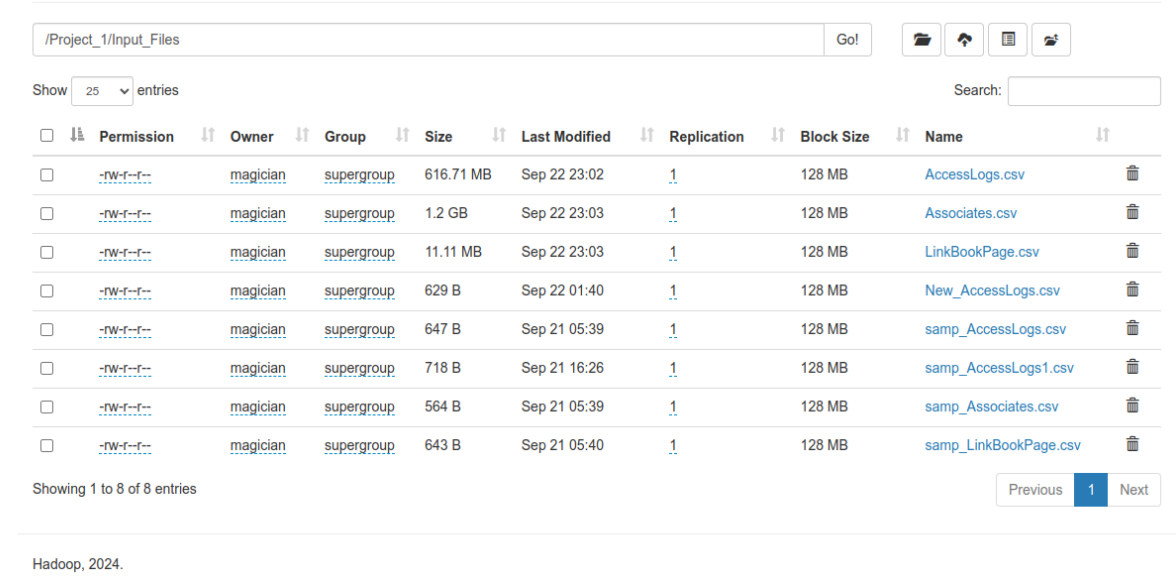
Here is the screenshot of the hadoop input folder and the datasets.




Main Project View




Two directories for input and output




Files inside Input directory

The huge files: AccessLogs.csv, Associates.csv and LinkBookPage.csv are the main files. The other files are the sample files to test our output.

# 3. Accomplishing Analytics Tasks using MapReduce Jobs [80 Points]

## Task a

Report the frequency of each education level (use HighestEdu as indication) on LinkBook.

### Assumptions/ Understading

From LinkBookPage, we have to get all the Highest Education column value, and map it to “one” in mapper phase, and then get the count of each education in reducer phase.

### Mapper/Reducer/Driver

**Mapper_1:** HighestEdu_Mapper.java

In this mapper phase we process each row as String. Using index position, we access the highest education column value (in our case, it is in index number 4). The value from highest education (BE, MS, PhD) will be mapped to one, and we will emit highest education as key, and one as a value.

Emitting Key-value pair: **<highest_education, one>**

**Reducer_1:** HighestEdu_Reducer.java

The output of Mapper_1 is passed as input for reducer, in which it calculates the count of highest education, and returns highest education and count.

Emitting Key-value pair: **<highest_education, count>**

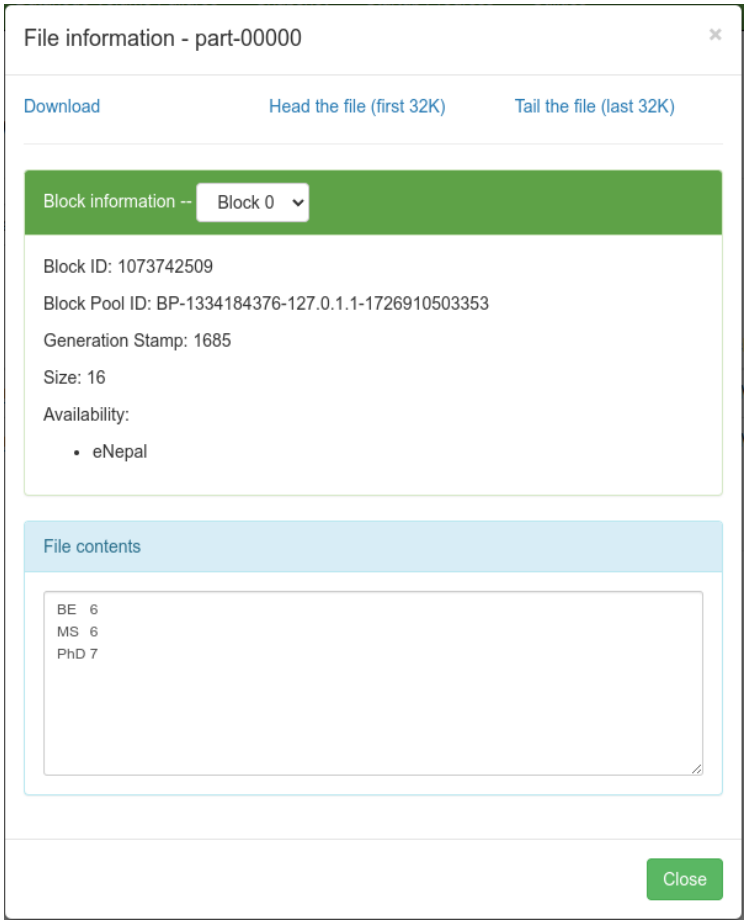
## Optimizations

We used combiner to write intermediate output of the mapper to the disk that is required for reducer, to reduce the input and output cost of the reducer.

## JAR Upload Command

magician@eNepal:~/IdeaProjects/TaskA$ hadoop jar /home/magician/IdeaProjects/TaskA/target/TaskA-1.0-SNAPSHOT.jar org.ujjwal.HighestEdu /Project_1/Input_Files/samp_LinkBookPage.csv /Project_1/Output_Files/TaskA_Output_Sample

## Output



# Task b

Find the 10 most popular LinkBook pages, namely, those that got the most accesses based on the AccessLog among all pages. Return Id, NickName, and Occupation.

## Assumptions/ Understading

From AccessLog, we are getting WhatPage Column values, to determine which Page has the most access by getting the count of Page Id, to emit Ids' respective name, and occupation from LinkBookPage.

## Mapper/Reducer/Driver

### **Mapper_1:** Mapper_1.java

In this mapper phase we process each row as String from AccessLog. Using index position, we access the WhatPage column value (in our case, it is in index number 2). The value from WhatPage will be mapped to one, and we will emit Page id as key, and one as a value.

Emitting Key-value pair: **<Page_id, one>**

### **Reducer_1:** CountReducer.java

The output of Mapper_1 is passed as input for reducer, in which it calculates the count of Page id, sort it in descending order and emit only top 10 records.

Emitting Key-value pair: **<Page_id, "">.**

### **Mapper_2:** Mapper_2.java

In this mapper phase, we process each row as String from LinkBookPage. Using index position, we emit ID as key, nickname and occupation as value (index position 0, 1, and 2 respectively, in our case.)

Emitting Key-value pair: **<Page_id, (nickname, occupation) >**

### **Reducer_1:** Final_Reducer.java

The output of Mapper_2 and CountReducer is passed as inputs for this reducer, in which it compares the id of both the inputs, and if it matches, it emits id, nickname and occupation. Since, reducer 1 will only pass top 10 values, we will have top user id, name, occupation.

Emitting Key-value pair: **<Page_id, (nickname, occupation) >.**

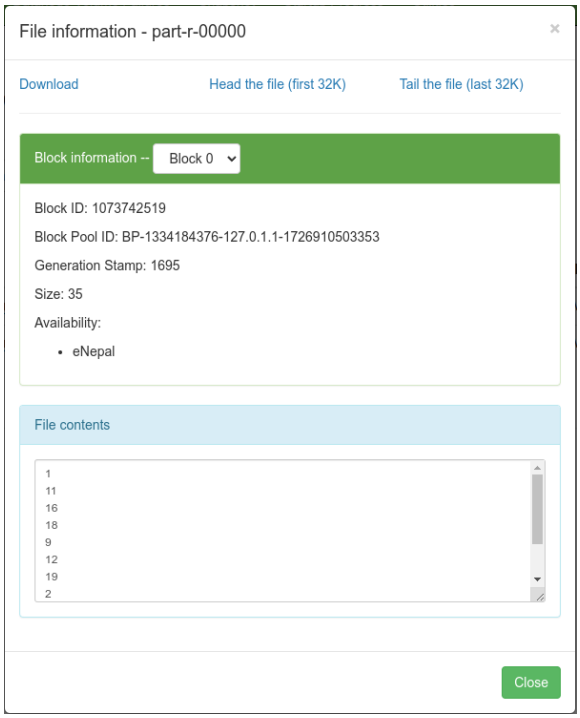
## Optimizations

- We used two combiners for Reducer_1 (CountReducer) and Reducer_2 (FinalReducer) to reduce the input/output cost of the reducer.
- To ensure the load is distributed efficiently, we used Job Chaining to reduce the size of dataset in each job. Having an itermediate output, keeps the job execution organized and easier to manage.

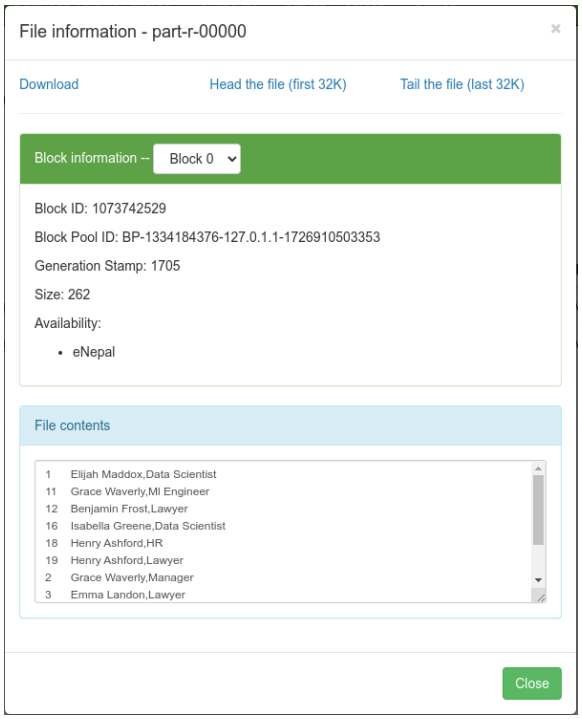
## JAR Upload Command

**magician@eNepal:~/IdeaProjects/TaskB$** hadoop jar /home/magician/IdeaProjects/TaskB/target/TaskB-1.0-SNAPSHOT.jar org.ujjwal.Driver /Project_1/Input_Files/New_AccessLogs.csv /Project_1/Output_Files/taskB_intermediate_output /Project_1/Input_Files/samp_LinkBookPage.csv /Project_1/Output_Files/taskB_output

# Output




Intermediate Output: that prints top 10 ids in sorted order.




Final Output: that prints the id, name, and occupation of the matching ids.

# Task c

Report all LinkBookPage users (NickName, and Occupation) whose highest education (HighestEdu) is the same as your own highest education (pick one). Note that the highest education in the data file may be random sequence of characters unless you work with meaningful strings like “Undergraduate” or “Graduate”. This is up to you.

## Assumptions/ Understading

We use LinkBookpage to get HighestEdu column value and matches it with degree(in our case MS).If it matches with this degree ,it will emit name and occupation.

## Mapper/Reducer/Driver

### Mapper_1: NameOcc_Mapper.java

In this mapper phase we process each row as String.In this we get HighestEdu which is at index position 4. If it matches with Ms degree then it will emit name as key and occupation as a value.

Emitting Key-value pair: **<nickname, occupation>**

## Optimizations

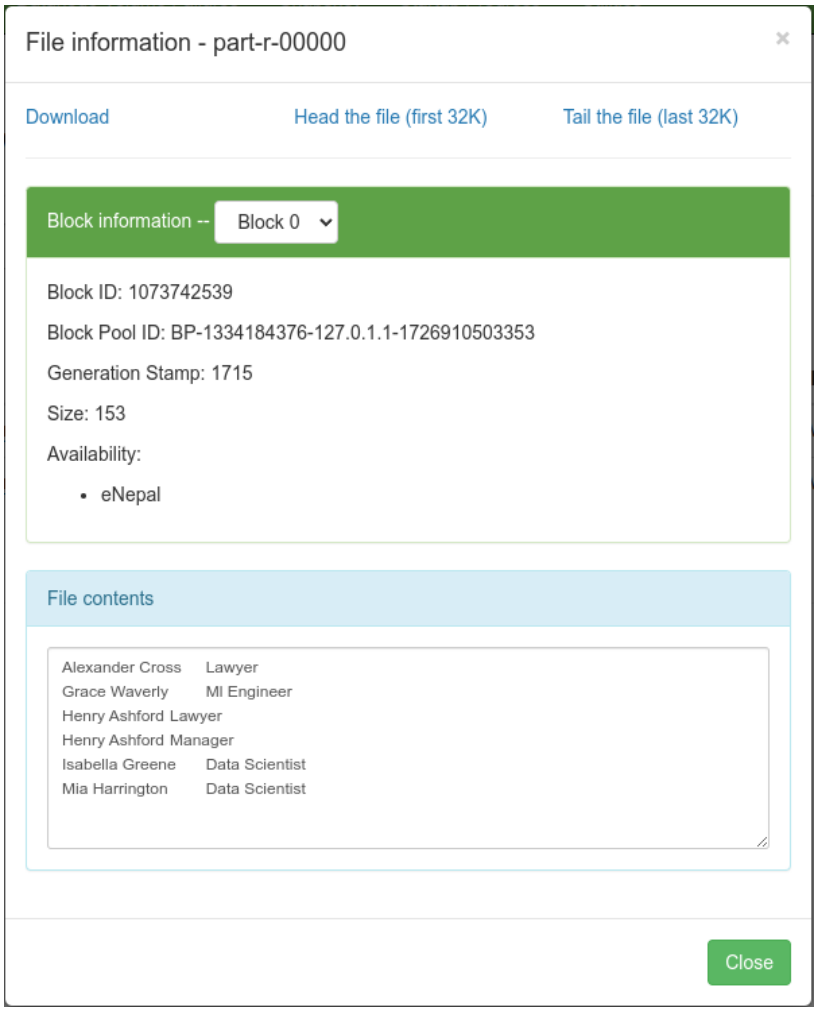
As this code need only Mapper phase, we didn’t use combiner class .

## JAR Upload Command

**magician@eNepal:~/IdeaProjects/TaskC$** hadoop jar /home/magician/IdeaProjects/TaskC/target/TaskC-1.0-SNAPSHOT.jar

org.ujjwal.NameOcc /Project_1/Input_Files/samp_LinkBookPage.csv /Project_1/Output_Files/TaskC_output

## Output



# Task d

For each LinkBookPage, compute the “happiness factor” of its owner. That is, for each LinkBookPage, report the owner’s nickname, and the number of relationships they have. For page owners that aren't listed in Associates, return a score of zero. Please note that we maintain a symmetric relationship, take that into account in your calculations.

## Assumptions/ Understading

From LinkBookPage we are getting id and nickname and from Associates we getting values of Id1 and Id2 column. In reducer we count the relation of each Id1(Count of relation is happyfactor).Finally,we are going to print the name and count.

## Mapper/Reducer/Driver

### **Mapper1:** LinkBookPageMapper.java

In this mapper phase we process each row as String and we are emitting id and nickname from LinkBookPage.

Emitting Key-value pair: **<id, name>**

### **Mapper2:** AssociatesMapper.java

In this mapper phase we process each row as String and we are emitting Value Id1 and Id2 seperately from Associates, with a dummy value.

Emitting Key-value pair: **<id1, “Dummy”>,<id2, “Dummy”>**

## Reducer1:

In this Reducer phase we are getting input from Mapper1 and Mapper2.
In this we get name from Mapper1 and store it in nicknamemap(Hashmap).Then get the Count of relationship of column Id1 from input of Mapper2 and store id and count in happinessmap(Hashmap).Finally, we aro going to emit the name and hapinessfactor of all the ids from Hashmap.
Emitting Key-value pair: **<id1, "Dummy">,<id2, "Dummy">**

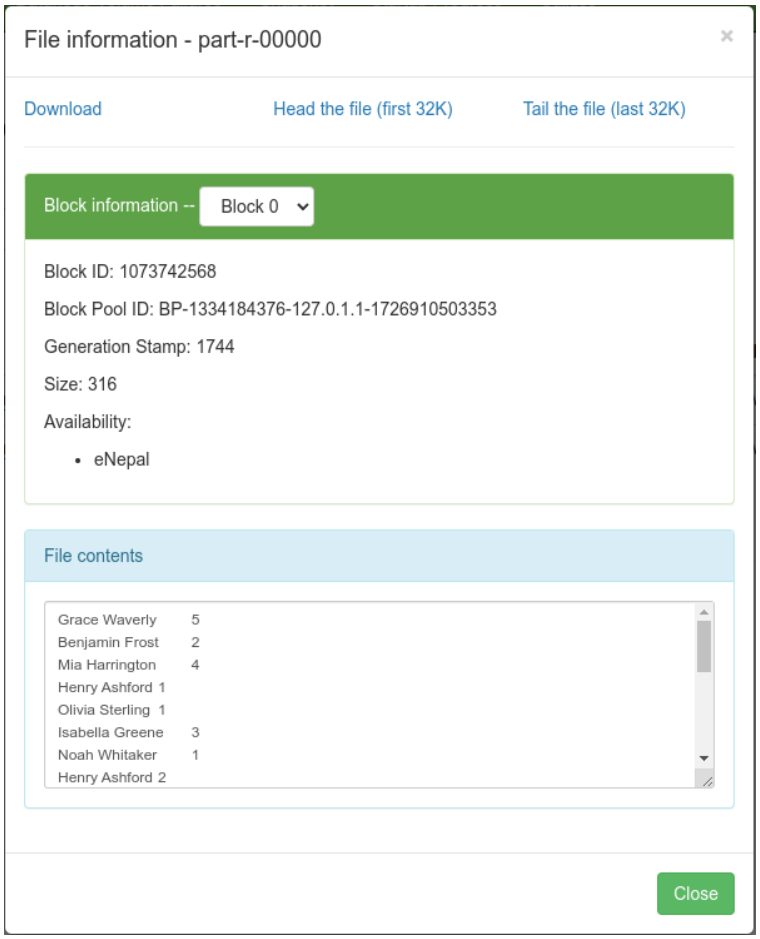
## Optimizations

- We used combiners for Reducer1 to reduce the input/output cost of the reducer.
- Used MultipleInput to minimize the job's runtime and to parallelly to different input files.
-Used cleanup method to reduce the number of writes during the reducer phase.

## JAR Upload Command

magician@eNepal:~/IdeaProjects/TaskD$ hadoop jar /home/magician/IdeaProjects/TaskD/target/TaskD-1.0-SNAPSHOT.jar org.ujjwal.HappyCount /Project_1/Input_Files/samp_LinkBookPage.csv /Project_1/Input_Files/samp_Associates.csv /Project_1/Output_Files/TaskD_output

# Output


File information - part-r-00000
Download
Head the file (first 32K)
Tail the file (last 32K)
Block information -- Block 0
Block ID: 1073742568
Block Pool ID: BP-1334184376-127.0.1.1-1726910503353
Generation Stamp: 1744
Size: 316
Availability:
eNepal
File contents
Grace Waverly 5
Benjamin Frost 2
Mia Harrington 4
Henry Ashford 1
Olivia Sterling 1
Isabella Greene 3
Noah Whitaker 1
Henry Ashford 2
Close

# Task e

Determine which people have favorites. That is, for each LinkBookPage owner, determine how many total accesses to LinkBookPage they have made (as reported in the AccessLog) and how many distinct LinkBookPage they have accessed in total. As for the identifier of each LinkBookPage owner, you don't have to report name. IDs are enough.

## Assumptions/ Understading

In this we are just gettin getting value of column Bywho and whatpage and we are to count the total number of pages visited and number of distinct page visited by an id(ByWho) and emits id ,total count and count of distinct page.

## Mapper/Reducer/Driver

### Mapper1: DistinctMapper.java

In this mapper phase we process each row as String and we are emiting id (Index:1) and pageid (Index:2) from Input file AccessLogs.

Emitting Key-value pair: **<id, pageid>**

### Reducer1: DistinctReducer.java

In this reducer phase we are going to caluclate the count of all the pages and distinct page visited by an id and finally emit the id ,total page count and distinct page count.

Emitting Key-value pair: **<id,(totalpagecount,distinctpagecount)>**

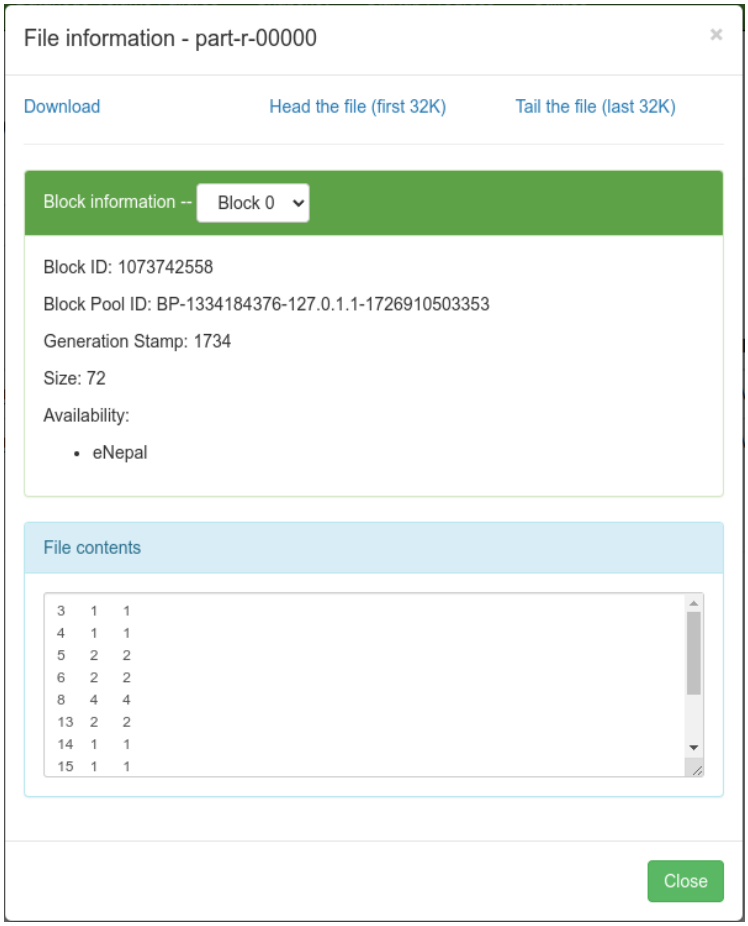
## Optimizations

- We used combiners for Reducer_1 to reduce the input/output cost of the reducer.

## JAR Upload Command

magician@eNepal:~/IdeaProjects/TaskE$ hadoop jar /home/magician/IdeaProjects/TaskE/target/TaskE-1.0-SNAPSHOT.jar org.ujjwal.DistinctPage /Project_1/Input_Files/samp_AccessLogs.csv /Project_1/Output_Files/TaskE_output

## Output



# Task f

Report all owners of a LinkBookPage who are more popular than an average user,namely, those who have more relationships than the average number of relationships across all owners LinkBookPages.

## Assumptions/ Understading

In this from two in input files Associates and LinkBookPage ,we are getting value of Id1and Id2 and id andname respective files.After that, we are going to find the average popularity and emit only name and popularity count of whose popularity count is higher than average popularity.

## Mapper/Reducer/Driver

### Mapper_1:AssocialtesMapper.java

In this mapper phase we process each row as String and value of column Id1 and Id2 as key and pass “Asso” (reference variable) and “1” as value (which will be converted to integer to get the relationship count).

Emitting Key-value pair: **<Id1,(Asso+’’,’’+ “1”)>,<Id1,(Asso+’’,’’+ “1”)>**

### Mapper_2:LinkBookpageMapper.java

In this mapper phase we process each row as String and value of column id and name(reference variable:”Link”) will be emitted.

Emitting Key-value pair: **<Id,”Link”+”,”+name>**

### Reducer1:ReduceJoinReducer.java

In this reducer phase, it has two inputs one from mapper 1 (to count relationship) and mapper 2 (emits id and name).Based on  from input1 we are going to find the average popularity,by finding the relationship count and divide by the total number of id (n:total number of id in our case) and if the popular count is

greater than the average popularity(1.9 in our case) and it emits the name and popularity count .

Emitting Key-value pair: **<name,popularity_count>**

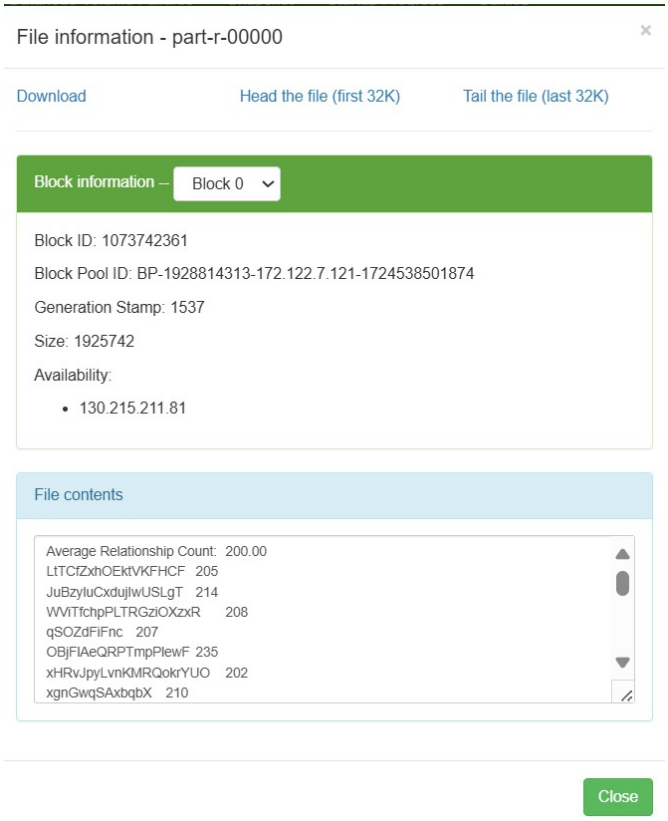
## Optimizations

- We used combiners for Reducer1 to reduce the input/output cost of the reducer.
- Used MultipleInput to minimize the job's runtime and to parallelly to different input files.
-Used cleanup method to reduce the number of writes during the reducer phase.

## JAR Upload Command

**magician@eNepal:~/IdeaProjects/TaskF$** hadoop jar /home/magician/IdeaProjects/TaskF/target/TaskF-1.0-SNAPSHOT.jar org.ujjwal.UserPopularityDriver /Project_1/Input_Files/samp_LinkBookPage.csv /Project_1/Input_Files/samp_Associates.csv /Project_1/Output_Files/TaskF_output

# Output




Output of the big data file.

# Task g

dentify "outdated" LinkBookPage. Return IDs and nicknames of persons that have not accessed LinkBook for 90 days (i.e., no entries in the AccessLog in the last 90 days).

## Assumptions/ Understading

In this we are going to get the column value of accestime (which is in minutes). Check whether the accesstime is greater than 90 days (129600 min).If it is greater, it is going to emit respective user's id and nickname.

## Mapper/Reducer/Driver

### **Mapper_1:** Mapper_1.java

In this mapper phase we process each row as String and get the id from index 1 and time from index 4 from AccessLogs and if time is greater than 129600 min, it is going to emit id and dummy value

Emitting Key-value pair: **<id,"one">**

### **Reducer_1:**Reducer_1.java

In this Reducer phase, the output from mapper 1 is taken as input and the count of each key is calculated and emiited

Emitting Key-value pair: **<id, id_count>**

### **Mapper_2:** Mapper_2.java

In this mapper phase we process each row as String.This mapper gets input from reducer 1 and emits the id as key and value as place holder.

Emitting Key-value pair: **<id, "">**

### **Mapper_3:** Mapper_3.java

In this mapper phase we process each row as String and get value of column Id and nickname from LinkBookPage and emit id as a key and name as value.

Emitting Key-value pair: **<id, name>**

### **Reducer_2:**FinalReducer.java

In this Reducer phase, the output from mapper 2 and 3 is taken as input and if the id matches it emits the id and name

Emitting Key-value pair: **<id, name>**

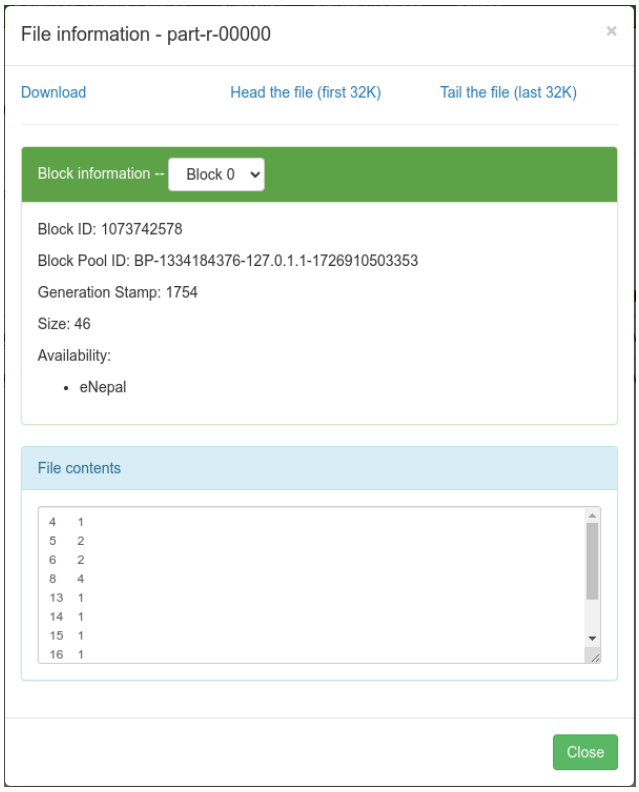
## Optimizations

- As there are 2 reducers so we are using 2 combiners for Reducer1 to reduce the input/output cost of the reducer.

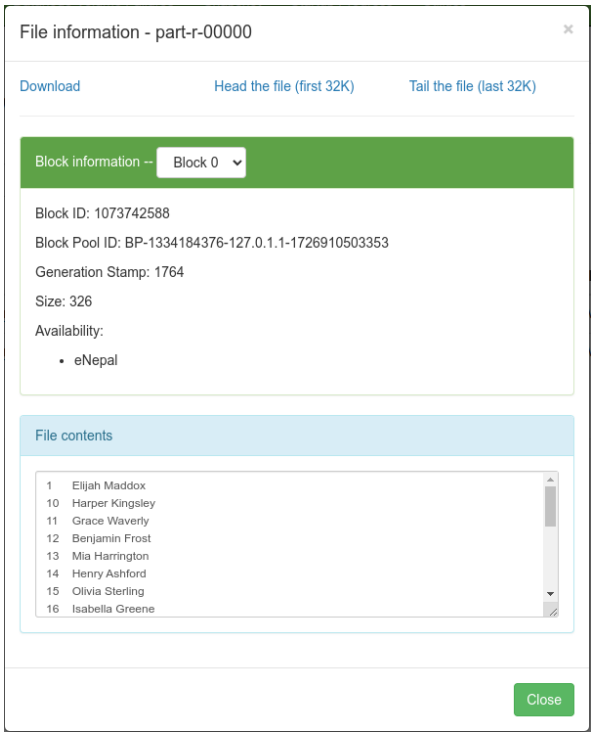
## JAR Upload Command

**magician@eNepal:~/IdeaProjects/TaskG$** hadoop jar /home/magician/IdeaProjects/TaskG/target/TaskG-1.0-SNAPSHOT.jar org.ujjwal.Outdated_driver /Project_1/Input_Files/samp_AccessLogs.csv /Project_1/Output_Files/TaskG_intermediate /Project_1/Input_Files/samp_LinkBookPage.csv /Project_1/Output_Files/TaskG_output

# Output




Intermediate Output:Id and Id_count




Final Output:Emits id and name

# Task h

Identify people that have a relationship with someone (Associates); yet never accessed their respective friend's LinkBookPage. Report IDs and nicknames.

## Assumptions/ Understading

To solve this question, We need to know the relationship of all the id of column Id1 and Id2 and vice versa.If the relationship exists we are going to check whether the id checked their related id's page .If not it is going to emit the id and nickname of who did'nt vist the related id's page.

## Mapper/Reducer/Driver

### **Mapper_1:** Mapper_1.java

In this mapper phase we process each row as String and gets id1 and id2 and emit id1 and id2, and id2 and id1 to know the relationship in both ways.

Emitting Key-value pair: **<Id1, Id2>,<Id2, Id1>**

### **Reducer_1:** Reducer_1.java

In this reducer phase we are going to find only unique id pairs by removing duplicates and emit the unique id pairs in both ways.

Emitting Key-value pair: **<Id1, Id2>,<Id2, Id1>**

### **Mapper_2:** Mapper_2.java

In this mapper phase we process each row as String and gets accessed user id and page owner id from AccessLogs and emit user id and page id.

Emitting Key-value pair: **<userId, pageid>**

### **Reducer_2:** Reducer_2.java

In this reducer phase we are going to find only unique user id and page id pairs by removing duplicates and emit the unique id pairs in both ways.

Emitting Key-value pair: **<userId, pageId>**

### **Mapper_3:** ComparisonMapper.java

This Mapper get the input  from reducer 1 and emits id .

Emitting Key-value pair: **<id1,id2>**

### **Reducer_3:** ComparisonReducer.java

In this reducer it gets input  id pair from reducer 1 and store it in hashmap.Then check whether the id visited their respective related id's page.if not it will emit the id of who did'nt access their related id's page and placeholde("") as a value.

Emitting Key-value pair: **<id, "">**

### **Mapper_3:** Mapper_3.java

This Mapper get value from column id and name from LinkBookPage and emit id as key and name as value.

Emitting Key-value pair: **<id,name>**

### **Reducer_3:** Reducer_3.java

This reducer gets input from reducer 2 and mapper 3 and emits id and name of only common ids in both the inputs.

Emitting Key-value pair: **<id,name>**

## **Optimizations**

-As there are 4 reducers so we are using 4 combiners for Reducer1 to reduce the input/output cost of the reducer.

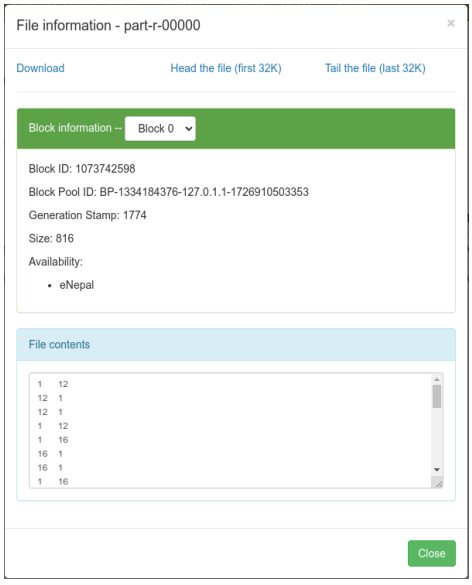
-- Used MultipleInput to minimize the  job's runtime  and to parallelly to different input files.

- To ensure the load is distributed efficiently, we used Job Chaining to reduce the size of dataset in each job. Having an itermediate output, keeps the job execution organized and easier to manage.

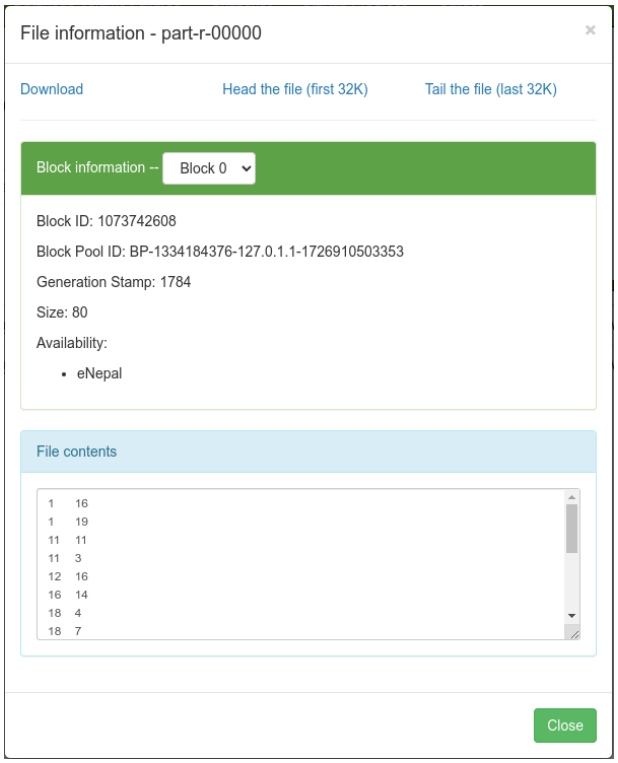
## JAR Upload Command

magician@eNepal:~/IdeaProjects/TaskH$ hadoop jar /home/magician/IdeaProjects/TaskH/target/TaskH-1.0-SNAPSHOT.jar org.ujjwal.MainDriver /Project_1/Output_Files/TaskH_intermediate1 /Project_1/Input_Files/samp_Associates.csv /Project_1/Output_Files/TaskH_intermediate2 /Project_1/Input_Files/New_AccessLogs.csv /Project_1/Output_Files/TaskH_intermediate3 /Project_1/Input_Files/samp_LinkBookPage.csv /Project_1/Output_Files/taskH_output

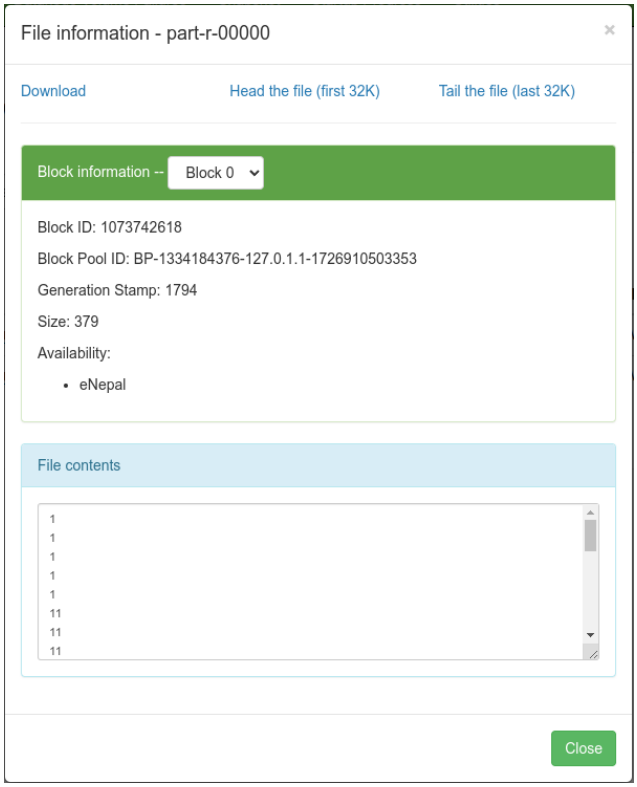
## Output



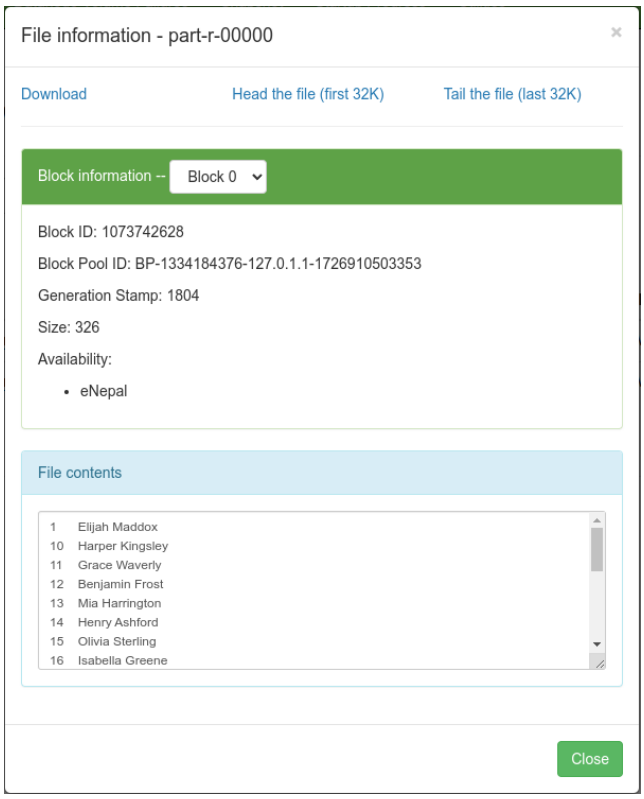

Intermediate Output 1:<Id1,Id2>




Intermediate Output 2:Unique id pair




Intermediate Output 3:




Final Outpu:id and name

# Teammate Contribution

Team Members : Hari Prasath KP, Ujjwal Pandit

Roles: We did everything together.

We had specified times on when we are going to work in hadoop. We did each and every step consulting with each other and working on it together.

The toughest part for us was setting the hadoop itself. We were trying to setup hadoop in our Windows machine, and we were not able to execute jar file, and were stuck on it. Later, we found out that due to Windows licensing issue on some Windows Laptop, we were not able to create it. Later, we decided we will install Linux, and installed Ubuntu in one of the laptop, and the execution was successful.

We will first discuss the question, figure out a way how to do it, discuss with each other on why this should be the optimal solution, and code it together. Doing this helped us to learn from each other, as well as work on problems without being stuck for long. We used generative AI for optimization, and finding out errors for the problem, but it seems like ChatGPT and other AI tools does not have good hadoop understading.