Project 1

Yu Zhai, Rahul Jakhmola

1. Creating Datasets

We generate all three datasets with python. All three datasets we generated are csv files.

1.1 MyPage

To generate this dataset, we import names package and pycountry package to randomly generate some names, nationalities and country codes. And also we create a csv file recording some hobbies.

```
ID,Name,Nationality,CountryCode,Hobby
1,Frank Huang,Costa Rica,53,Inline skating
2,Shawna Gaddy,Switzerland,42,Cabaret
3,Joyce Mills,Wallis and Futuna,244,Baseball
4,Larry Perez,Grenada,91,Board sports
5,Debra Scoggins,Kazakhstan,117,Kabaddi
6,Kristin Nelson,Saint Helena Ascension and Tristan da Cunha,197,Poi
7,Annette Davis,Afghanistan,2,Pet
8,Richard Burlock,Morocco,138,Basketball
9,George Michaud,Italy,112,Kayaking
10,Johnny Fletcher,Saint Lucia,129,Jigsaw puzzles
11,Frederic Morales,Argentina,9,Mountain biking
12,May Royster,Saint Martin (French part),137,Dance
13,Jason Thier,Uruguay,234,Skiing
14,Michael Adkinson,Bahrain,25,Handball
15,Leopoldo Coleman,Greece,90,Geocaching
16,Wanda Mckenney,Bouvet Island,37,Stone skipping
17,Ross Hart,Micronesia Federated States of,78,Knapping
18,Greg Wooten,Equatorial Guinea,89,Worldbuilding
19,Shirley Fitch,Antarctica,12,Writing
20,Melissa Arnold,Isle of Man,104,Rowing
21,Matthew Tomasi,Pakistan,173,Poi
22,Sylvia Christy,Azerbaijan,17,Knapping
23,Maureen Rohling,Brunei Darussalam,35,Flying disc
24,Carrie Womack,Chile,43,Running
```

Above is how our dataset looks like.

1.2 AllFriends

We randomly pick one ID as PersonID and another ID as MyFriend, and make sure that these two IDs are not the same. We randomly generate a number between 1 and 1000000 for DateofFriendship.

```
relationships = ['collegefriend', 'girlfriend', 'boyfriend', 'family',
'highschoolfriend', 'childhoodfriend', 'colleague']
```

We also create a list to store different values of Desc.

```
FriendRel,PersonID,MyFriend,DateofFriendship,Desc
1,129219,34423,860326,boyfriend
2,165666,145389,372421,family
3,12376,198625,712329,girlfriend
4,162914,52505,797978,girlfriend
5,170290,121203,733383,boyfriend
6,119265,5508,861792,boyfriend
7,69404,102155,622560,collegefriend
8,95355,139082,305568,childhoodfriend
9,36127,160913,635104,highschoolfriend
10,30308,46587,989867,girlfriend
11,124362,123989,12378,highschoolfriend
12,8915,55730,272144,highschoolfriend
13,29345,190099,246525,family
14,94139,31782,13081,highschoolfriend
15,120247,43440,70316,girlfriend
16,156008,644,437134,boyfriend
17,27925,127336,416515,highschoolfriend
18,167158,21946,141262,colleague
19,50280,86749,312277,girlfriend
20,64151,51865,337356,colleague
21,92114,34868,101788,colleague
22,185864,22217,54073,colleague
23,114642,132399,77742,collegefriend
24,183473,139247,673108,colleague
25,15440,96127,163407,family
```

This is how our dataset looks like.

1.3 AccessLog

The way to generate this dataset is similar to last one. We randomly generate two IDs for ByWho and WhatPage columns.

```
accesses = ['just viewed', 'left a note', 'added a friendship']
```

We create a list to store the values for accesses type.

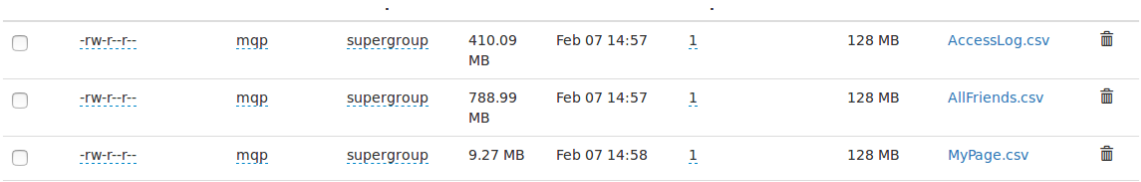
```
AccessID,ByWho,WhatPage,TypeOfAccess,AccessTime
1,191067,157236,left a note,571248
2,118213,12390,left a note,791252
3,86013,167348,just viewed,75284
4,119793,114884,left a note,200759
5,159511,66211,left a note,670480
6,8550,111372,added a friendship,810857
7,82543,136231,just viewed,708615
8,134881,162544,left a note,672763
9,123909,175403,just viewed,195862
10,114003,11360,just viewed,710705
```

```
11,91887,30425,added a friendship,513875
12,128611,169023,added a friendship,597768
13,180707,35266,left a note,208834
14,83939,22829,added a friendship,373913
15,372,24199,just viewed,920349
16,77920,181257,added a friendship,683410
17,70953,56811,left a note,149418
18,104150,155326,just viewed,455265
19,196888,175175,added a friendship,514305
20,126808,133920,left a note,997560
21,189913,111066,just viewed,848295
22,113840,51510,added a friendship,27128
23,42632,197369,left a note,303747
24,84119,172707,added a friendship,531181
```

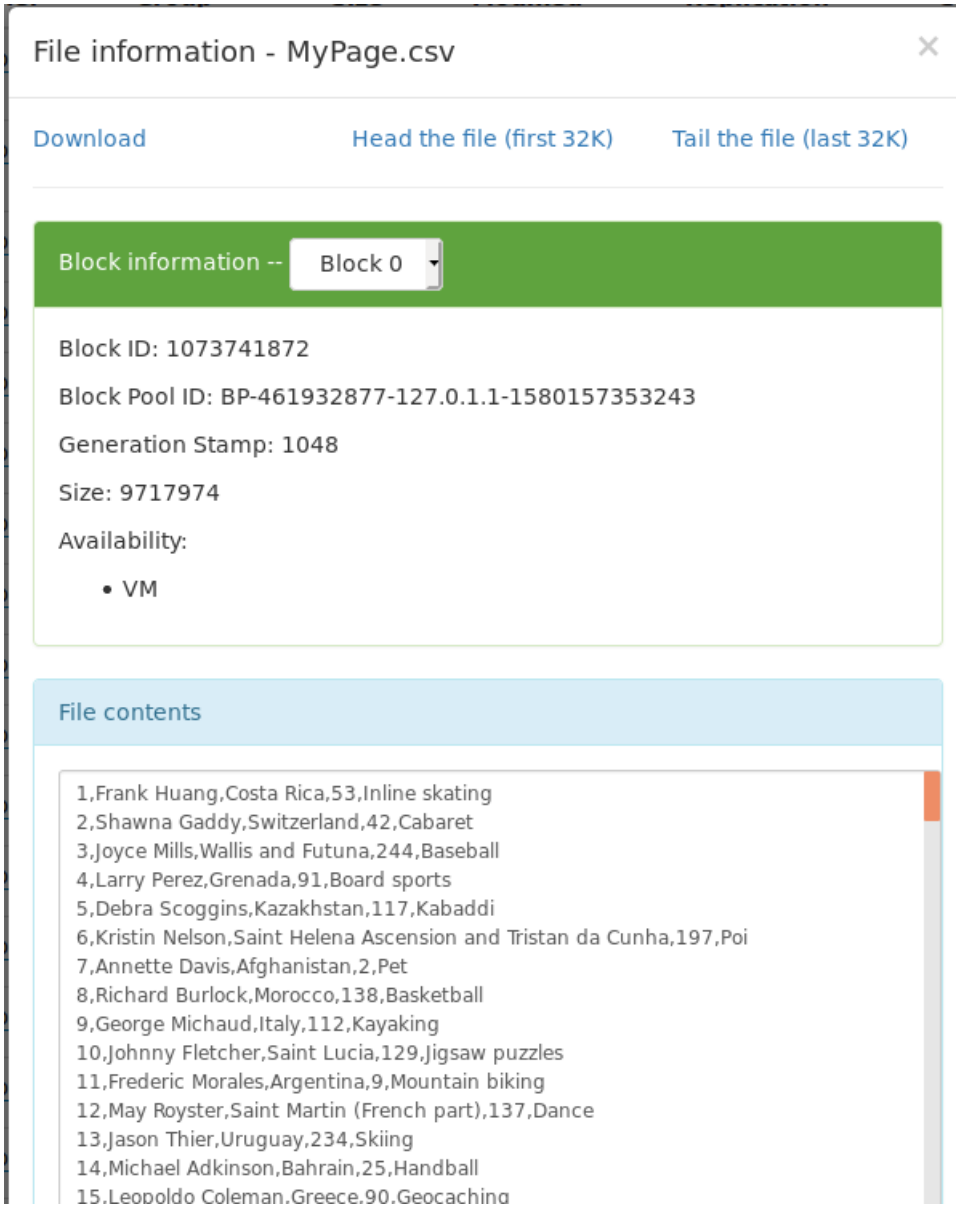
This is how our dataset looks like.

2. Loading Datasets into Hadoop

We use the command **hadoop fs -put localfile** to upload all three datasets to the HDFS.



☐	-rw-r--r--	mgp	supergroup	410.09 MB	Feb 07 14:57	1	128 MB	AccessLog.csv
☐	-rw-r--r--	mgp	supergroup	788.99 MB	Feb 07 14:57	1	128 MB	AllFriends.csv
☐	-rw-r--r--	mgp	supergroup	9.27 MB	Feb 07 14:58	1	128 MB	MyPage.csv



3. Solutions

3.1 Task a

We only create a mapper for filtering. In the mapper, we write out all the users' nationalities that are same as mine, which is China.

```
Theresa Vincent,Nordic skating
Cecilia Berczel,Horseback riding
Colleen Evanich,tabletop games
Chris Rosales,Shopping
Alfonso Griffin,Gambling
Sara Shankle,Model building
Daniel Eaves,Hooping
Michael Furnas,Kitesurfing
Nancy Huddleston,Geocaching
Judith Benson,Dance
Victor Gonzales,Baseball
Jeffrey Croft,Backpacking
Angela Mariska,Skateboarding
Matthew Harris,Homebrewing
Dorothy Powell,Archery
Claude Hebert,Coloring
Donna Morris,Cabaret
Bianca Sansone Knapping
```

This is our results. To make sure that this is the right answer, we also write out their nationalities.

```
Theresa Vincent,Nordic skating,China
Cecilia Berczel,Horseback riding,China
Colleen Evanich,tabletop games,China
Chris Rosales,Shopping,China
Alfonso Griffin,Gambling,China
Sara Shankle,Model building,China
Daniel Eaves,Hooping,China
Michael Furnas,Kitesurfing,China
Nancy Huddleston,Geocaching,China
Judith Benson,Dance,China
Victor Gonzales,Baseball,China
Jeffrey Croft,Backpacking,China
Angela Mariska,Skateboarding,China
Matthew Harris,Homebrewing,China
Dorothy Powell,Archery,China
Claude Hebert,Coloring,China
Donna Morris,Cabaret,China
Bianca Sansone Knapping China
```

3.2 Task b

We create the mapper to write out the key-value pair: [nationality, 1]. And in the reducer, we will sum all them up to get the number of users in different countries. We also set the combiner, the function is same as the reducer.

File contents

```
Afghanistan     825
Albania     820
Algeria     792
American Samoa      792
Andorra     824
Angola      785
Anguilla    820
Antarctica      840
Antigua and Barbuda     840
Argentina 831
Armenia     780
Aruba       850
Australia   784
Austria     730
Azerbaijan      809
Bahamas  755
Bahrain     756
Bangladesh      816
```

3.2 Task c

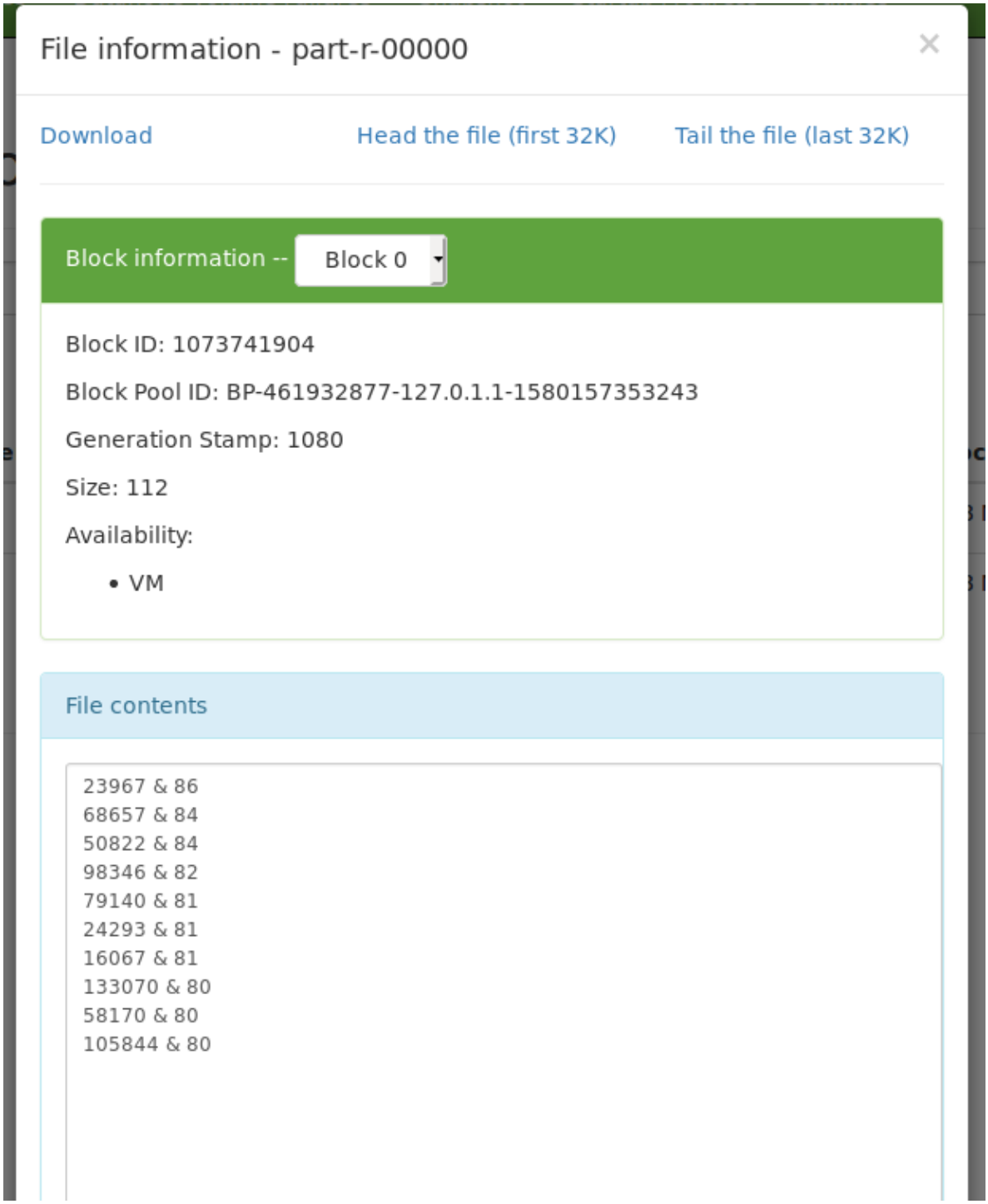
We create two jobs to finish this task. The first one is to get the number of accesses for every pages. It is same as task b. So the result of this job will be [pageID, # of accesses].

Then we create the second job. In the mapper of this job, we output the same result as the first job, but conversely. Which means the output of the mapper will be like [# of accesses, pageID].

In the Hadoop, mapper will pass the record to the reducer in the ascending order of the keys. We override the comparator function in hadoop to make the mapper pass the record in descending order.

```
public static class KeyComparator extends WritableComparator{
        protected KeyComparator(){
            super(IntWritable.class, true);
        }
        public int compare(WritableComparable a, WritableComparable b)
{
            return -super.compare(a, b);
        }
    }
```

Then in the reducer, we just need to set a variable N = 0. Once we output a record from reducer, N will increase by 1. Once the N = 10, it means that we have already output the 10 pages that has the most accesses.



3.4 task d

We create 2 jobs to finish this task. The first job is to calculate the number of friends each user has. In the mapper, we write out the key-value pair [MyFriend, 1]. Then in the reducer, we sum them up, it will get the number of people who list the user as friend.

In the mapper of the second job, it will output the key-value pair [ID, info]. Info may be the user's name or the number of his/her friends. Then in the reducer, we just need to check the first character of the info. If the character is digital, it represents the number of friends. Otherwise, it represents the user's name. If the info is the user's name, we will set it as key, otherwise we will set it as value. Then the reducer will write out the key-value pair, which represents the name and the number of friends.

File contents

```
Frank Huang    99
Johnny Fletcher        102
Donna Carper   83
Karen Paredes  114
Michael Malik  97
Jesse Swanger  109
Donovan Evans          99
Kim Johnson    105
Justina Matelski       97
Laura Popp     104
Kerri Gautreau 120
Thelma Harris  102
Marlene Martinez       100
John Mclain    102
James Cross    100
```

3.5 task e

In the mapper, we output the key-value pair [ID, access]. ID means the ID of the person who has accessed the Facebook page. Access means the Id of the page that was accessed.

In the reducer, we set a variable **sum** to record the total accesses to Facebook pages they have made. Every time the reducer get a record, the sum will add 1. And we also set a hash table to store the ID of the page that was accessed. So in the end of reducer, we just need to output the size of the hash table to get how many distinct Facebook pages they have accessed in total.

File contents

```
1    6    5
10   7    5
100  6    6
1000      7    5
10000     6    4
100000    5    5
100001    7    7
100002    6    4
100003    7    7
100004    6    5
100005    5    5
100006    6    6
100007    5    5
100008    5    5
100009    4    4
```

3.6 task f

For this task, we have two different understanding, thus we have two different solutions.

Solution1： In the mapper, we output the key-value pair [ID, time]. ID means the ID of the person who has accessed the Facebook page. Time references to the access time.

In the reducer, we set a variable **max** to record the maximum access time for each user. If the maximum time is less than 800000, we will write out the ID.

File contents

```
108491
12653
156886
165685
184880
2775
33769
34146
46864
718
```

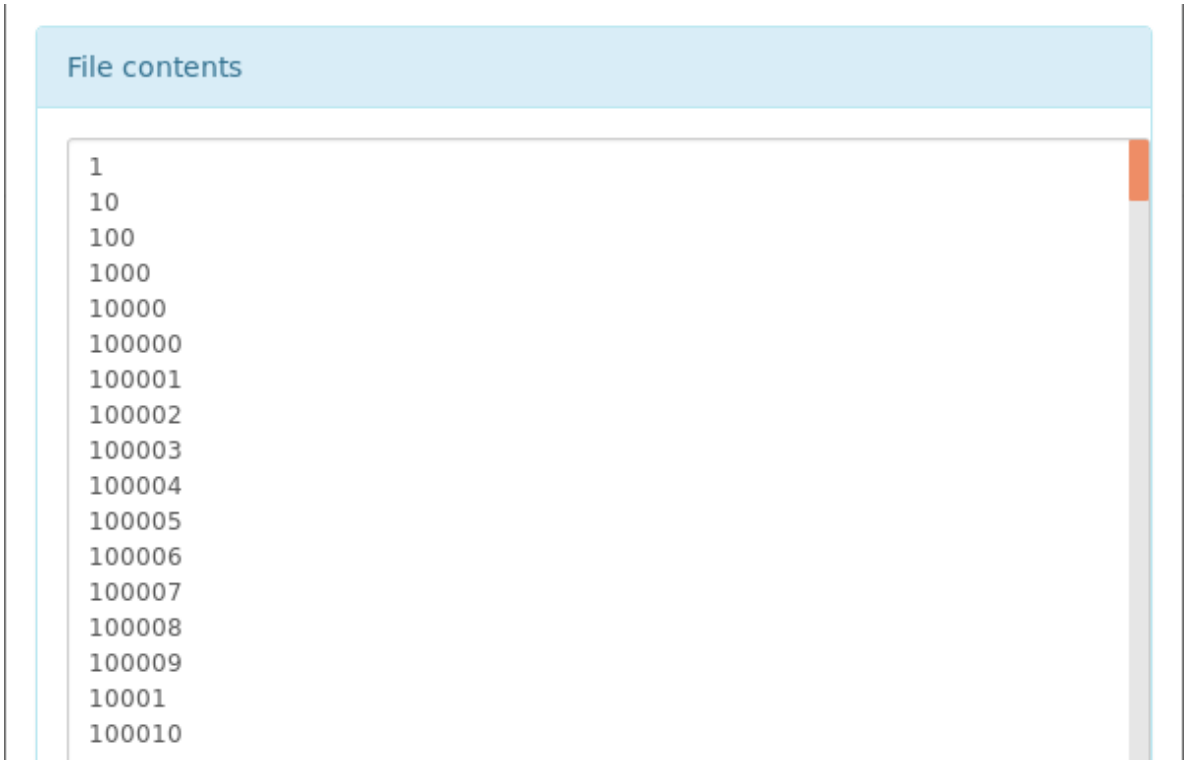
Solution2:We define tenure as the time elapsed from first access to the last access. Then, people whose tenure is shorter than a threshold are said to have expired accounts.

3.7 task g

We create two mappers for two different input files. The input file of the first mapper will be AccessLog. And it will output the key-value pair [ID, access]. ID references to the ID of the person who has accessed the Facebook page. Access references to the ID of the page that was accessed. And we add a 'A' in the front of friend, so the reducer can recognize it.

The input file of the second mapper will be AllFriends. It will output the key-value pair [ID, friend]. ID references to the Person-ID of the user. Friend references to ID of a person that the user are friend with. And we add a 'F' in the front of friend, so the reducer can recognize it.

In the reducer, we create two hash table. One is to store the ID of the user's friends. The other one is to store the ID of the page that the user accessed. Then we just need to check if all the records in the friends hash table are in the access hash table. If not, we will write out the ID of the user.



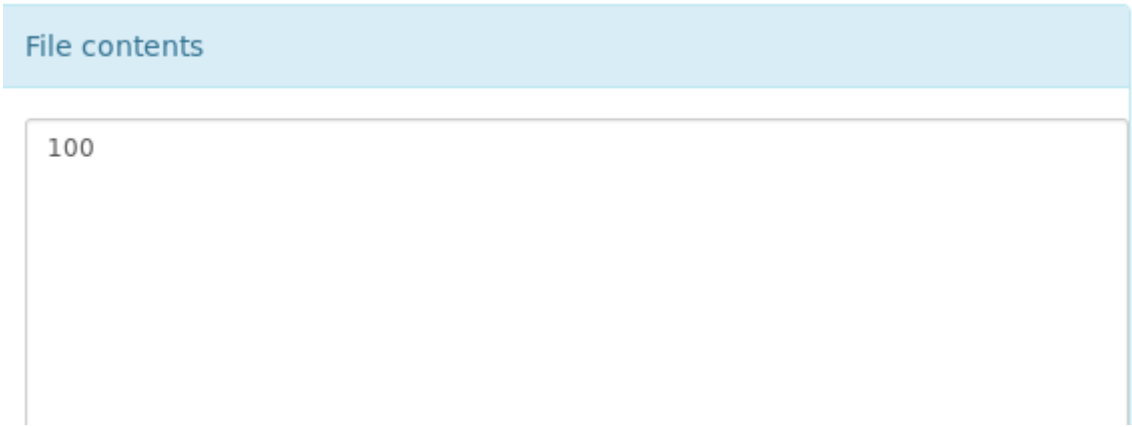
And the result is quiet interesting, cause we found out that all the users have friends that then don't care.

3.8 task h

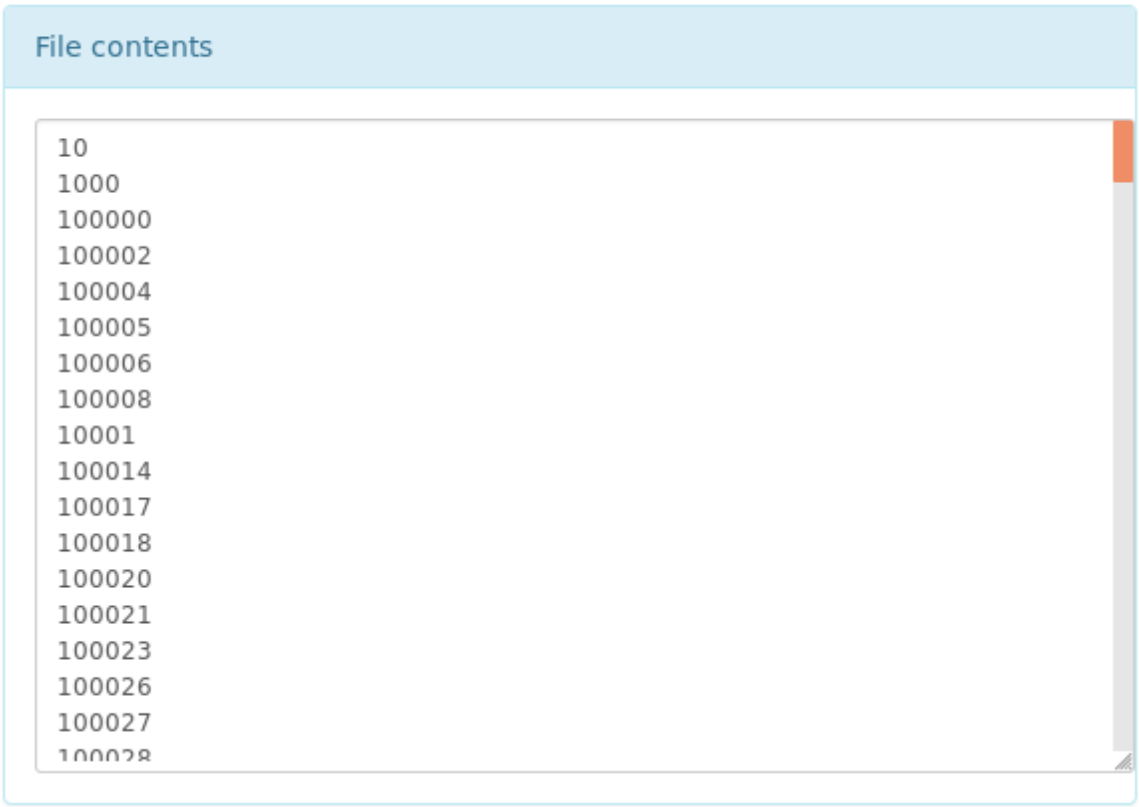
In this task, we use the result from the first step in task d. The result store the number of friends of each user.

Then we create a job to calculate the mean value. In the mapper, we write out the key-value pair [1, # of friends]. The reason that we set the key to 1 is to make sure all the records have the same key, so we can get the total number of friends that all the users have in the reducer.

In the reducer, we set a variable **num** to record the number of users we have. Every time the reducer receive a record, the **num** will increase by 1. And we sum all the values up, and divided it by **num**, then we got the mean value. The mean value is 100.



In the second job, we just need a mapper. The input file will be the csv file contains the number of friends each user has. So in the mapper, we just need to check whether the number of friends the user have is greater than 100. If yes, we write out the ID.



Contribution

Yu: Generate the AllFriends and AccessLog datasets. MapReduce part.(50%)

Rahul: Generate the MyPage dataset. Pig Part.(50%)