

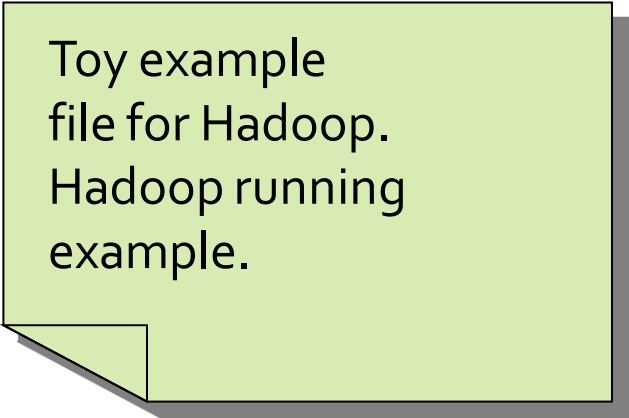
MapReduce - Exercises

Exercise #1

- Word count problem
 - Input: (unstructured) textual file
 - Output: number of occurrences of each word appearing in the input file

Exercise #1 - Example

- Input file



Toy example
file for Hadoop.
Hadoop running
example.

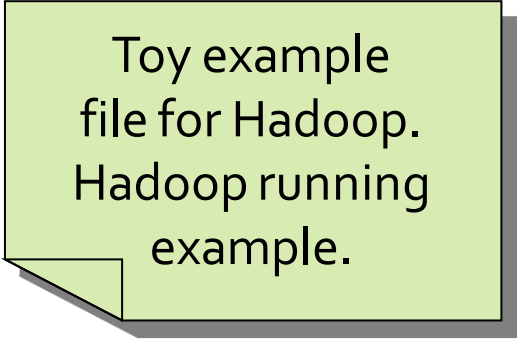
- Output pairs (toy, 1)
(example, 2)
(file, 1)
(for, 1)
(hadoop, 2)
(running, 1)

Exercise #2

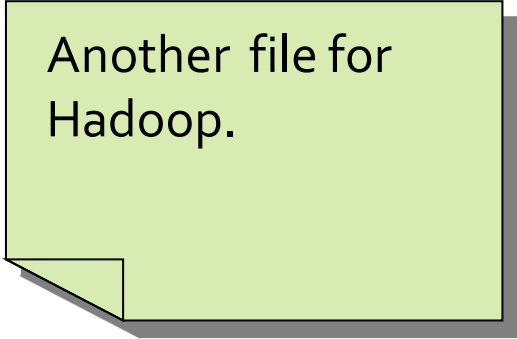
- Word count problem
 - Input: a HDFS folder containing textual files
 - Output: number of occurrences of each word appearing in at least one file of the collection (i.e., files of the input directory)
- The only difference with respect to exercise #1 is given by the input
 - Now the input is a collection of textual files

Exercise #2 - Example

- Input files



Toy example
file for Hadoop.
Hadoop running
example.



Another file for
Hadoop.

- Output pairs (another, 1)
(example, 2)
(file, 2)
(for, 2)
(hadoop, 3)
(running, 1)
(toy, 1)

Exercise #3

- PM₁₀ pollution analysis
 - Input: a (structured) textual file containing the daily value of PM₁₀ for a set of sensors
 - Each line of the file has the following format
sensorId,date\tPM₁₀ value (μg/m³)\n
 - Output: report for each sensor the number of days with PM₁₀ above a specific threshold
 - Suppose to set threshold = 50 μg/m³

Exercise #3 - Example

- Input file

s1,2016-01-01	20.5
s2,2016-01-01	30.1
s1,2016-01-02	60.2
s2,2016-01-02	20.4
s1,2016-01-03	55.5
s2,2016-01-03	52.5

- Output pairs (s1, 2)
(s2, 1)

Exercise #4

- PM₁₀ pollution analysis per city zone
- Input: a (structured) textual file containing the daily value of PM₁₀ for a set of city zones
 - Each line of the file has the following format
zoneId,date\tPM₁₀ value ($\mu\text{g}/\text{m}^3$)\n
- Output: report for each zone the list of dates associated with a PM₁₀ value above a specific threshold
 - Suppose to set threshold = $50 \mu\text{g}/\text{m}^3$

Exercise #4 - Example

- Input file

zone1,2016-01-01	20.5
zone2,2016-01-01	30.1
zone1,2016-01-02	60.2
zone2,2016-01-02	20.4
zone1,2016-01-03	55.5
zone2,2016-01-03	52.5

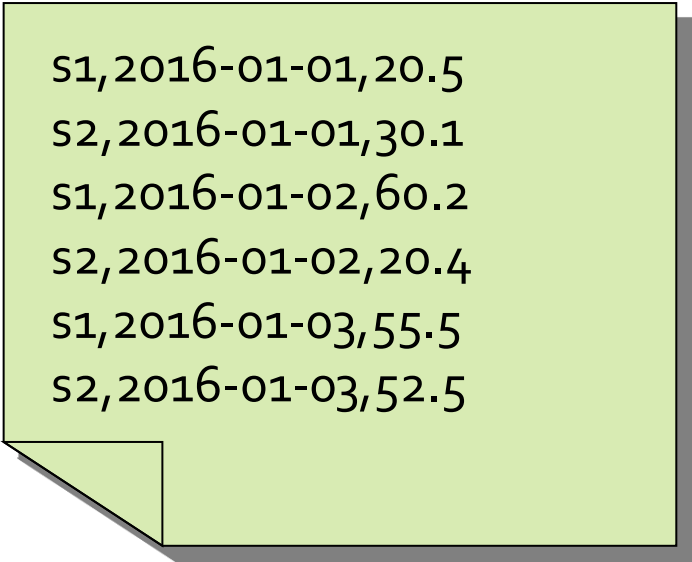
- Output pairs (zone1, [2016-01-03, 2016-01-02])
(zone2, [2016-01-01])

Exercise #5

- Average
 - Input: a collection of (structured) textual csv files containing the daily value of PM₁₀ for a set of sensors
 - Each line of the files has the following format
sensorId,date,PM₁₀ value (µg/m³)\n
 - Output: report for each sensor the average value of PM₁₀

Exercise #5 - Example

- Input file



```
s1,2016-01-01,20.5  
s2,2016-01-01,30.1  
s1,2016-01-02,60.2  
s2,2016-01-02,20.4  
s1,2016-01-03,55.5  
s2,2016-01-03,52.5
```

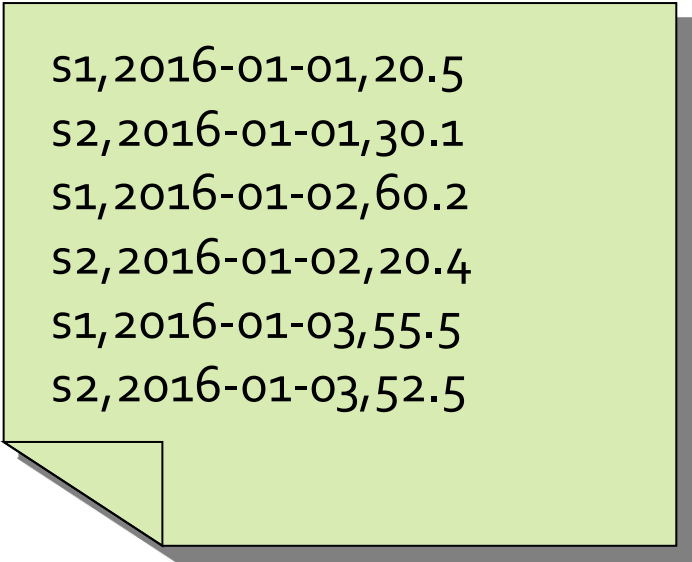
- Output pairs (s1, 45.4)
(s2, 34.3)

Exercise #6

- Max and Min
 - Input: a collection of (structured) textual csv files containing the daily value of PM₁₀ for a set of sensors
 - Each line of the files has the following format
sensorId,date,PM₁₀ value (μg/m³)\n
 - Output: report for each sensor the maximum and the minimum value of PM₁₀

Exercise #6 - Example

- Input file



```
s1,2016-01-01,20.5  
s2,2016-01-01,30.1  
s1,2016-01-02,60.2  
s2,2016-01-02,20.4  
s1,2016-01-03,55.5  
s2,2016-01-03,52.5
```

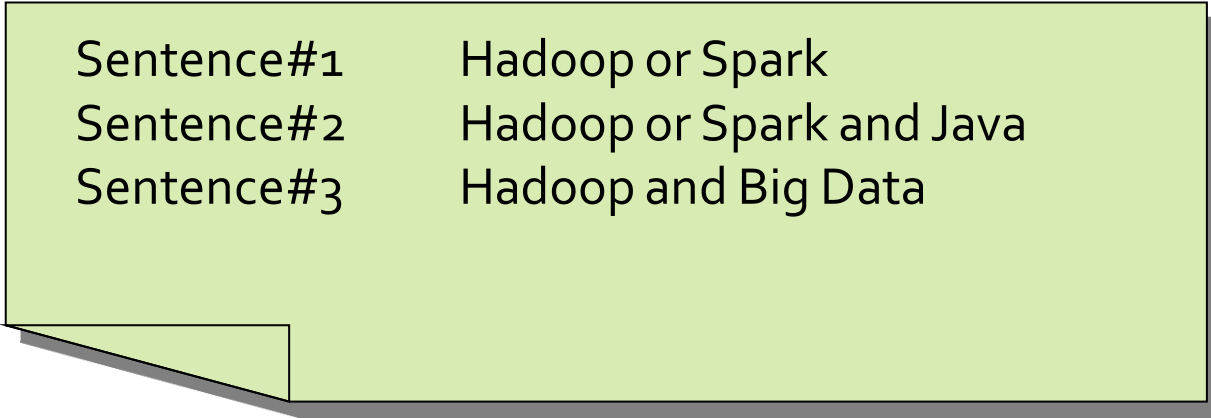
- Output pairs (s1, max=60.2_min=20.5)
(s2, max=52.5_min=20.4)

Exercise #7

- Inverted index
 - Input: a textual file containing a set of sentences
 - Each line of the file has the following format
sentenceld\tsentence\n
 - Output: report for each word **w** the list of sentencelds of the sentences containing **w**
 - Do not consider the words “and”, “or”, “not”

Exercise #7 - Example

- Input file



Sentence#1	Hadoop or Spark
Sentence#2	Hadoop or Spark and Java
Sentence#3	Hadoop and Big Data

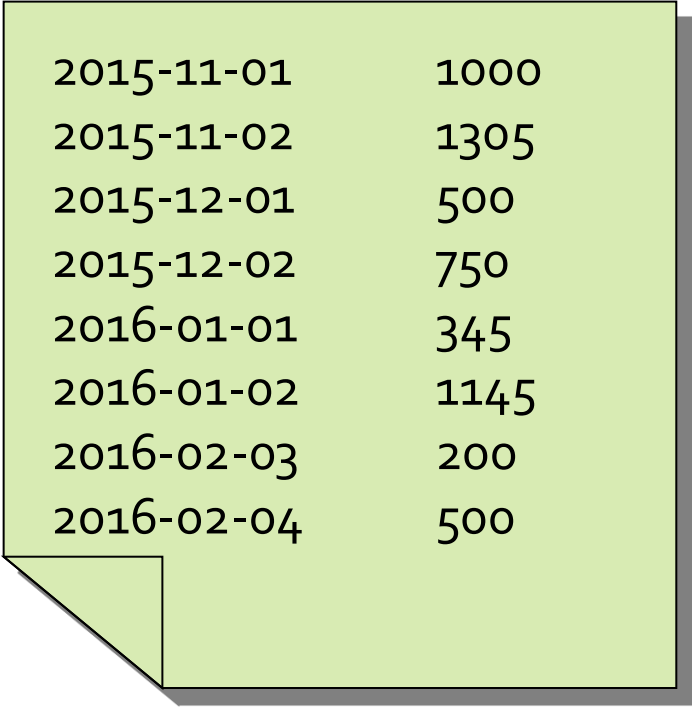
- Output pairs (hadoop, [Sentence#1, Sentence#2, Sentence#3])
(spark, [Sentence#1, Sentence#2])
(java, [Sentence#2])
(big, [Sentence#3])
(data, [Sentence#3])

Exercise #8

- Total income for each month of the year and Average monthly income per year
 - Input: a (structured) textual csv files containing the daily income of a company
 - Each line of the files has the following format
date\tdaily income\n
 - Output:
 - Total income for each month of the year
 - Average monthly income for each year

Exercise #8 - Example

- Input file



2015-11-01	1000
2015-11-02	1305
2015-12-01	500
2015-12-02	750
2016-01-01	345
2016-01-02	1145
2016-02-03	200
2016-02-04	500

- Output

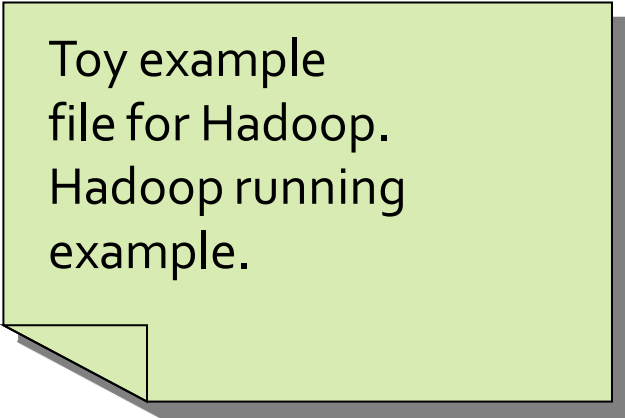
(2015-11, 2305)	(2015, 1777.5)
(2015-12, 1250)	
(2016-01, 1490)	(2016, 1095.0)
(2016-02, 700)	

Exercise #9

- Word count problem
 - Input: (unstructured) textual file
 - Output: number of occurrences of each word appearing in the input file
- Solve the problem by using in-mapper combiners

Exercise #9 - Example

- Input file



Toy example
file for Hadoop.
Hadoop running
example.

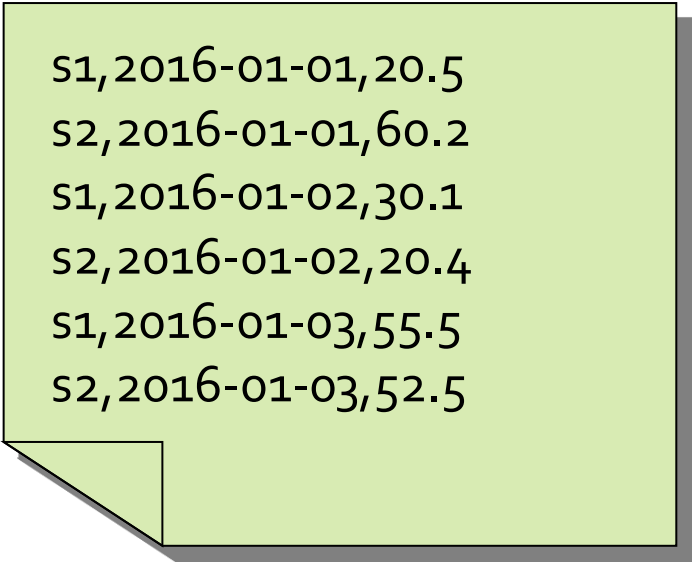
- Output pairs (toy, 1)
(example, 2)
(file, 1)
(for, 1)
(hadoop, 2)
(running, 1)

Exercise #10

- Total count
 - Input: a collection of (structured) textual csv files containing the daily value of PM₁₀ for a set of sensors
 - Each line of the files has the following format
sensorId,date,PM₁₀ value (μg/m³)\n
 - Output: total number of records

Exercise #10 - Example

- Input file



```
s1,2016-01-01,20.5  
s2,2016-01-01,60.2  
s1,2016-01-02,30.1  
s2,2016-01-02,20.4  
s1,2016-01-03,55.5  
s2,2016-01-03,52.5
```

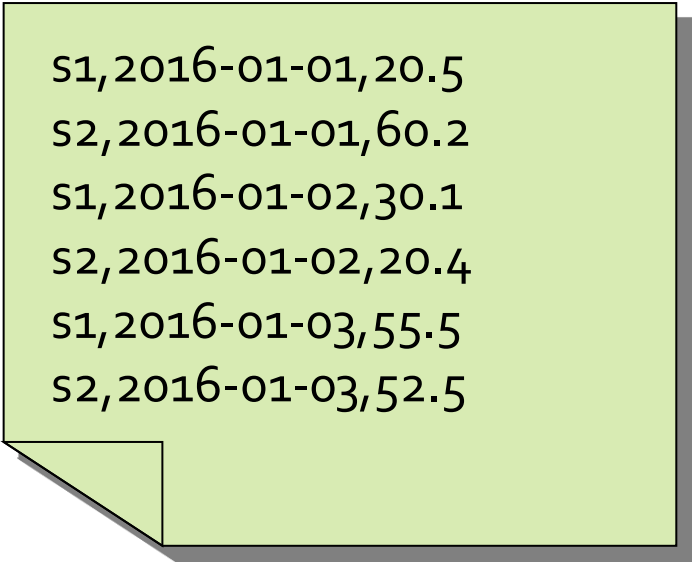
- Output: 6

Exercise #11

- Average
 - Input: a collection of (structured) textual csv files containing the daily value of PM₁₀ for a set of sensors
 - Each line of the files has the following format
sensorId,date,PM10 value ($\mu\text{g}/\text{m}^3$)\n
 - Output: report for each sensor the average value of PM₁₀
 - Suppose the number of sensors is equal to 2 and their ids are s1 and s2

Exercise #11 - Example

- Input file



```
s1,2016-01-01,20.5  
s2,2016-01-01,60.2  
s1,2016-01-02,30.1  
s2,2016-01-02,20.4  
s1,2016-01-03,55.5  
s2,2016-01-03,52.5
```

- Output

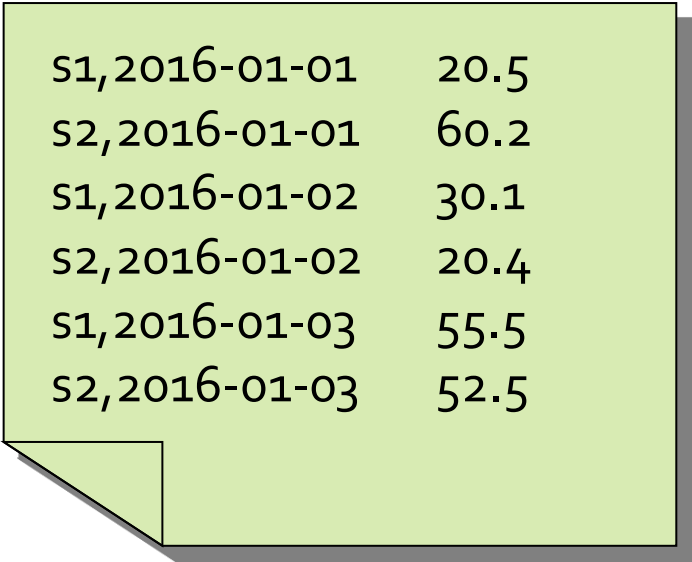
```
s1, 45.4  
s2, 34.3
```

Exercise #12

- Select outliers
 - Input: a collection of (structured) textual files containing the daily value of PM₁₀ for a set of sensors
 - Each line of the files has the following format
sensorId,date\tPM₁₀ value (µg/m³)\n
 - Output: the records with a PM₁₀ value below a user provided threshold (the threshold is an argument of the program)

Exercise #12 - Example

- Input file



s1,2016-01-01	20.5
s2,2016-01-01	60.2
s1,2016-01-02	30.1
s2,2016-01-02	20.4
s1,2016-01-03	55.5
s2,2016-01-03	52.5

- Threshold: 21

- Output

s1,2016-01-01	20.5
s2,2016-01-02	20.4

Exercise #13

- Top 1 most profitable date
 - Input: a (structured) textual csv files containing the daily income of a company
 - Each line of the files has the following format
date\tdaily income\n
 - Output:
 - Select the date and income of the top 1 most profitable date
 - In case of tie, select the first date

Exercise #13 - Example

- Input file

2015-11-01	1000
2015-11-02	1305
2015-12-01	500
2015-12-02	750
2016-01-01	345
2016-01-02	1145
2016-02-03	200
2016-02-04	500

- Output

2015-11-02	1305
------------	------

Exercise #13 Bis

- Top 2 most profitable dates
 - Input: a (structured) textual csv files containing the daily income of a company
 - Each line of the files has the following format
date\tdaily income\n
 - Output:
 - Select the date and income of the top 2 most profitable dates
 - In case of tie, select the first 2 dates among the ones associated with the highest income

Exercise #13 Bis - Example

- Input file

2015-11-01	1000
2015-11-02	1305
2015-12-01	500
2015-12-02	750
2016-01-01	345
2016-01-02	1145
2016-02-03	200
2016-02-04	500

- Output

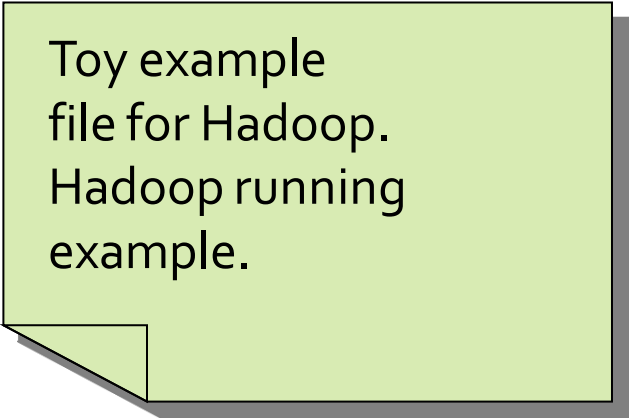
2015-11-02	1305
2016-01-02	1145

Exercise #14

- Dictionary
 - Input: a collection of news (textual files)
 - Output:
 - List of distinct words occurring in the collection

Exercise #14 - Example

- Input file



Toy example
file for Hadoop.
Hadoop running
example.

- Output

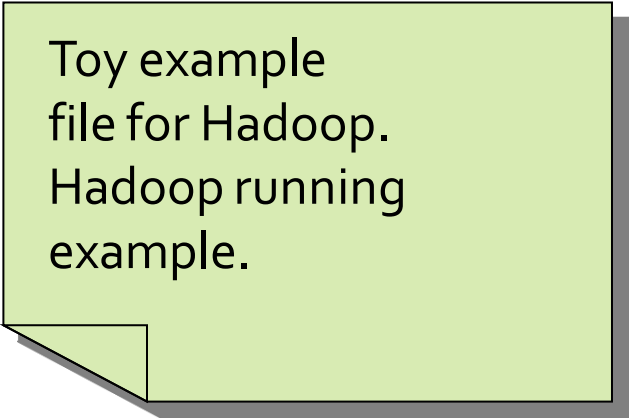
example
file
for
hadoop
running
toy

Exercise #15

- Dictionary – Mapping word - integer
 - Input: a collection of news (textual files)
 - Output:
 - List of distinct words occurring in the collection associated with a set of unique integers
 - Each word is associated with a unique integer (and viceversa)

Exercise #15 - Example

- Input file



Toy example
file for Hadoop.
Hadoop running
example.

- Output

(example, 1)
(file, 2)
(for, 3)
(hadoop, 4)
(running, 5)
(toy, 6)

Exercise #17

- Select maximum temperature for each date
 - Input: two structured textual files containing the temperatures gathered by a set of sensors
 - Each line of the first file has the following format
sensorID,date,hour,temperature\n
 - Each line of the second file has the following format
date,hour,temperature,sensorID\n
 - Output: the maximum temperature for each date (considering the data of both input files)

Exercise #17 - Example

■ Input files

```
s1,2016-01-01,14:00,20.5  
s2,2016-01-01,14:00,30.2  
s1,2016-01-02,14:10,11.5  
s2,2016-01-02,14:10,30.2
```

```
2016-01-01,14:00,20.1,s3  
2016-01-01,14:00,10.2,s4  
2016-01-02,14:15,31.5,s3  
2016-01-02,14:15,20.2,s4
```

■ Output

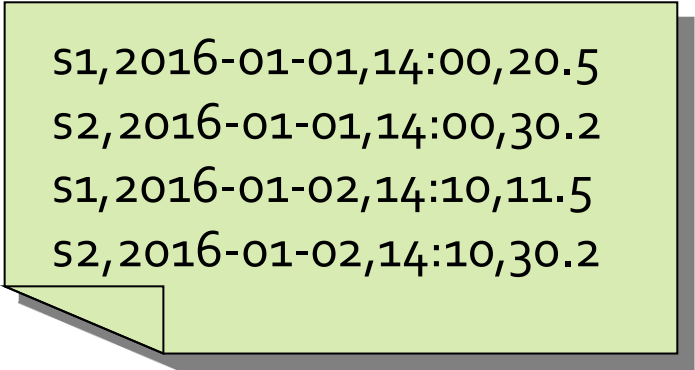
2016-01-01	30.2
2016-01-02	31.5

Exercise #18

- Filter the readings of a set of sensors based on the value of the measurement
 - Input: a set of textual files containing the temperatures gathered by a set of sensors
 - Each line of the files has the following format
sensorID,date,hour,temperature\n
 - Output:
 - The lines of the input files associated with a temperature value greater than 30.0

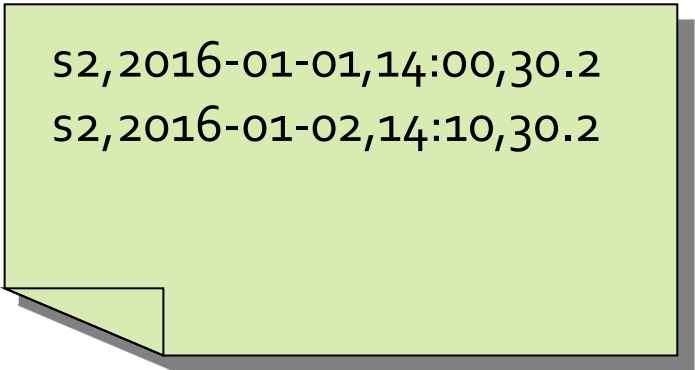
Exercise #18 - Example

- Input file



```
s1,2016-01-01,14:00,20.5  
s2,2016-01-01,14:00,30.2  
s1,2016-01-02,14:10,11.5  
s2,2016-01-02,14:10,30.2
```

- Output file



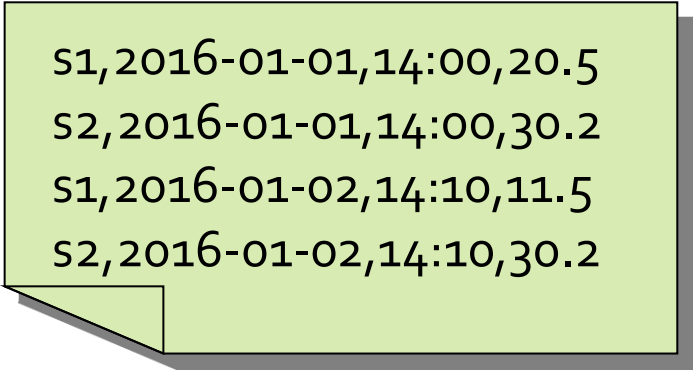
```
s2,2016-01-01,14:00,30.2  
s2,2016-01-02,14:10,30.2
```

Exercise #19

- Filter the readings of a set of sensors based on the value of the measurement
 - Input: a set of textual files containing the temperatures gathered by a set of sensors
 - Each line of the files has the following format
sensorID,date,hour,temperature\n
 - Output:
 - The lines of the input files associated with a temperature value less than or equal to 30.0

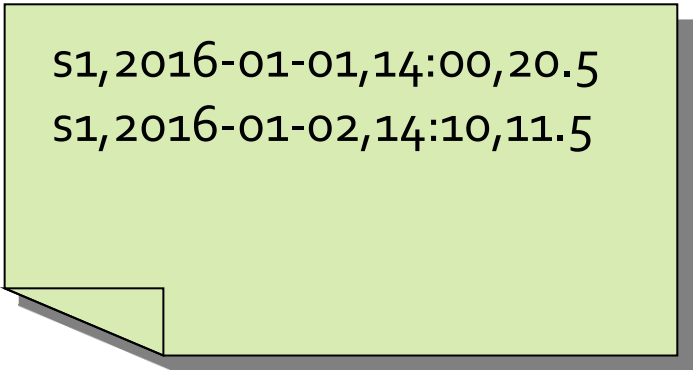
Exercise #19 - Example

- Input file



```
s1,2016-01-01,14:00,20.5  
s2,2016-01-01,14:00,30.2  
s1,2016-01-02,14:10,11.5  
s2,2016-01-02,14:10,30.2
```

- Output file



```
s1,2016-01-01,14:00,20.5  
s1,2016-01-02,14:10,11.5
```

Exercise #20

- Split the readings of a set of sensors based on the value of the measurement
 - Input: a set of textual files containing the temperatures gathered by a set of sensors
 - Each line of the files has the following format
sensorID,date,hour,temperature\n
 - Output:
 - a set of files with the prefix "high-temp-" containing the lines of the input files with a temperature value greater than 30.0
 - a set of files with the prefix "normal-temp-" containing the lines of the input files with a temperature value less than or equal to 30.0

Exercise #20 - Example

- Input file

```
s1,2016-01-01,14:00,20.5  
s2,2016-01-01,14:00,30.2  
s1,2016-01-02,14:10,11.5  
s2,2016-01-02,14:10,30.2
```

- Output files

high-temp-m-00001

```
s2,2016-01-01,14:00,30.2  
s2,2016-01-02,14:10,30.2
```

normal-temp-m-00001

```
s1,2016-01-01,14:00,20.5  
s1,2016-01-02,14:10,11.5
```

Exercise #20 Bis

- Split the readings of a set of sensors based on the value of the measurement
 - Input: a set of textual files containing the temperatures gathered by a set of sensors
 - Each line of the files has the following format
sensorID,date,hour,temperature\n
 - Output:
 - a set of files with the prefix "high-temp-" containing the temperatures associated with the lines of the input files with temperature values greater than 30.0
 - a set of files with the prefix "normal-temp-" containing the lines of the input files with a temperature value less than or equal to 30.0

Exercise #20 Bis - Example

- Input file

```
s1,2016-01-01,14:00,20.5  
s2,2016-01-01,14:00,30.2  
s1,2016-01-02,14:10,11.5  
s2,2016-01-02,14:10,41.5
```

- Output files

high-temp-m-00001

```
30.2  
41.5
```

normal-temp-m-00001

```
s1,2016-01-01,14:00,20.5  
s1,2016-01-02,14:10,11.5
```

Exercise #21

- Stopword elimination problem
 - Input:
 - A large textual file containing one sentence per line
 - A small file containing a set of stopwords
 - One stopword per line
 - Output:
 - A textual file containing the same sentences of the large input file without the words appearing in the small file
 - The order of the sentences in the output file can be different from the order of the sentences in the input file

Exercise #21 - Example

- Input files

- Large file

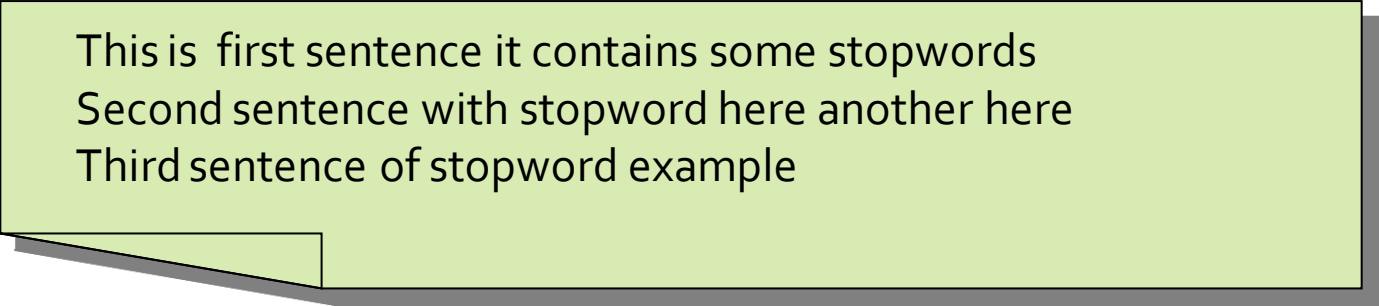
This is **the** first sentence **and** it contains some stopwords
Second sentence with **a** stopword here **and** another here
Third sentence of **the** stopword example

- Stopword file

a
an
and
the

Exercise #21 - Example

- Output file



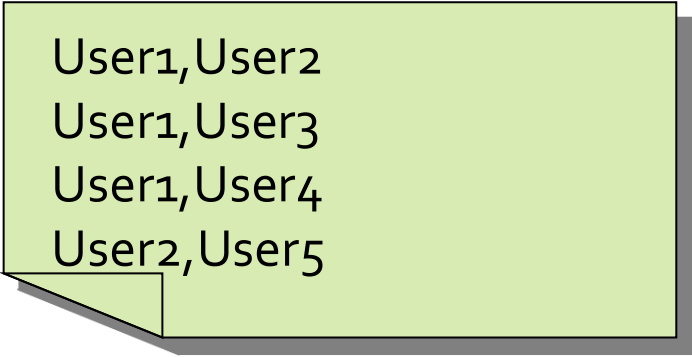
This is first sentence it contains some stopwords
Second sentence with stopword here another here
Third sentence of stopword example

Exercise #22

- Friends of a specific user
 - Input:
 - A textual file containing pairs of users (one pair per line)
 - Each line has the format
 - Username1,Username2
 - Each pair represents the fact that Username1 is friend of Username2 (and vice versa)
 - One username specified as parameter by means of the command line
 - Output:
 - The friends of the specified username stored in a textual file
 - One single line with the list of friends

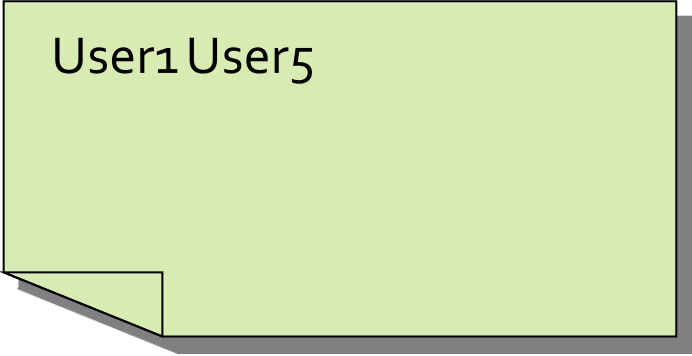
Exercise #22 - Example

- Input file



```
User1,User2  
User1,User3  
User1,User4  
User2,User5
```

- Username parameter: User2
- Output file



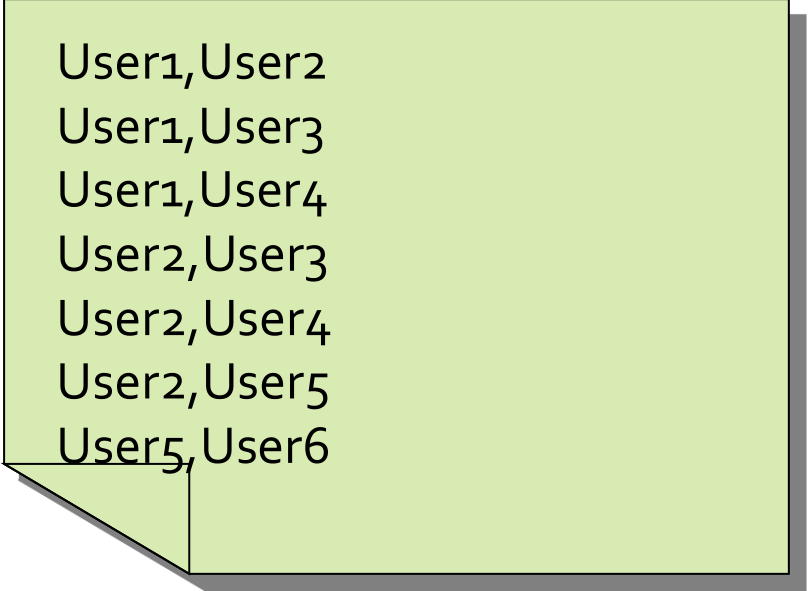
```
User1 User5
```


Exercise #23

- Potential friends of a specific user
 - Input:
 - A textual file containing pairs of users (one pair per line)
 - Each line has the format
 - Username1,Username2
 - Each pair represents the fact that Username1 is friend of Username2 (and vice versa)
 - One username specified as parameter by means of the command line
 - Output:
 - The potential friends of the specified username stored in a textual file
 - One single line with the list of potential friends
 - User1 is a potential friend of User2 if they have at least one friend in common

Exercise #23 - Example

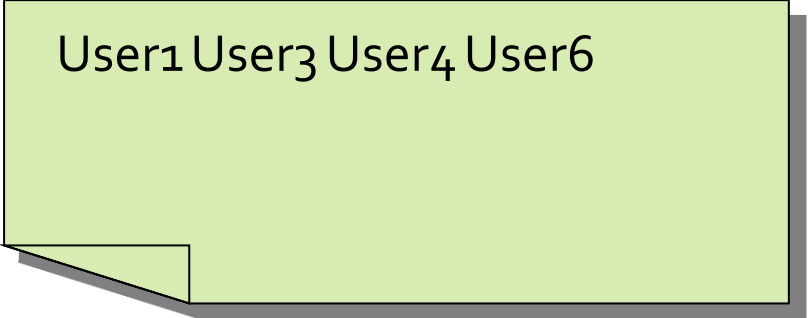
- Input file



```
User1,User2  
User1,User3  
User1,User4  
User2,User3  
User2,User4  
User2,User5  
User5,User6
```

- Username parameter: User2

- Output file



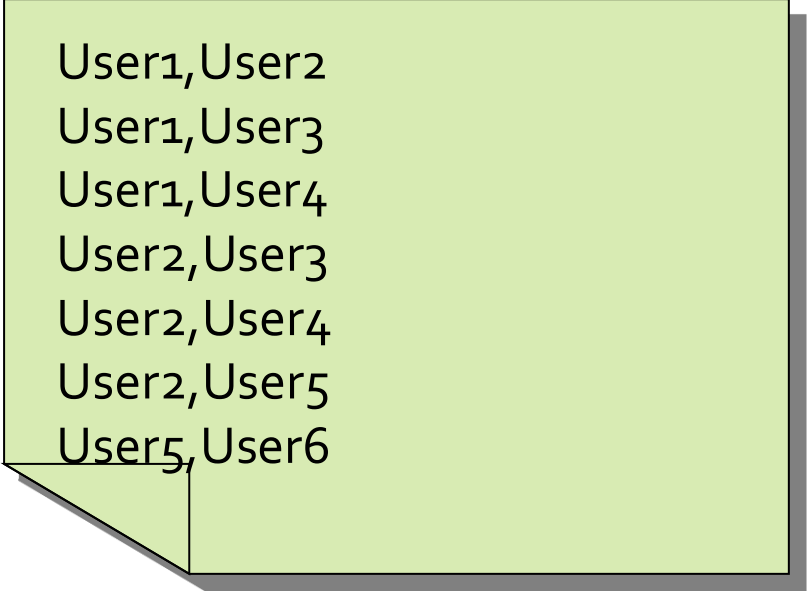
```
User1 User3 User4 User6
```

Exercise #23 Bis

- Potential friends of a specific user
 - Solve problem #23 by removing the friends of the specified user from the list of its potential friends

Exercise #23 Bis - Example

- Input file



```
User1,User2  
User1,User3  
User1,User4  
User2,User3  
User2,User4  
User2,User5  
User5,User6
```

- Username parameter: User2

- Output file



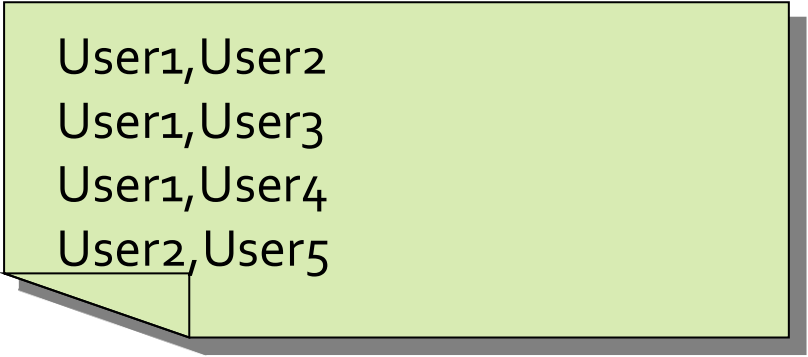
```
User6
```

Exercise #24

- Compute the list of friends for each user
 - Input:
 - A textual file containing pairs of users (one pair per line)
 - Each line has the format
 - Username1,Username2
 - Each pair represents the fact that Username1 is friend of Username2 (and vice versa)
 - Output:
 - A textual file containing one line for each user. Each line contains a user and the list of its friends

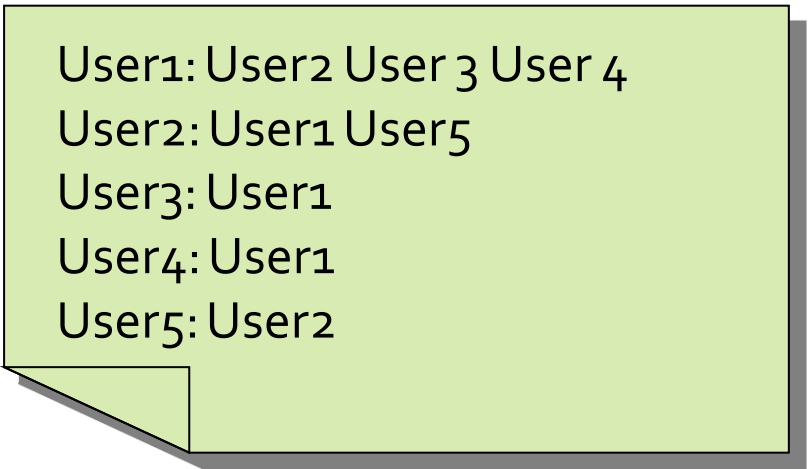
Exercise #24 - Example

- Input file



```
User1,User2  
User1,User3  
User1,User4  
User2,User5
```

- Output file



```
User1: User2 User 3 User 4  
User2: User1 User5  
User3: User1  
User4: User1  
User5: User2
```

Exercise #25

- Compute the list of potential friends for each user
 - Input:
 - A textual file containing pairs of users (one pair per line)
 - Each line has the format
 - Username1,Username2
 - Each pair represents the fact that Username1 is friend of Username2 (and vice versa)
 - Output:
 - A textual file containing one line for each user with at least one potential friend. Each line contains a user and the list of its potential friends
 - User1 is a potential friend of User2 if they have at least one friend in common

Exercise #25 - Example

- Input file

```
User1,User2  
User1,User3  
User1,User4  
User2,User3  
User2,User4  
User2,User5  
User5,User6
```

- Output file

```
User1: User2 User3 User4 User5  
User2: User1 User3 User4 User6  
User3: User1 User2 User4 User5  
User4: User1 User2 User3 User5  
User5: User1 User3 User4  
User6: User2
```


Exercise #26

- Word (string) to integer conversion
 - Input:
 - A large textual file containing a list of words per line
 - The small file dictionary.txt containing the mapping of each possible word appearing in the first file with an integer. Each line contain the mapping of a word with an integer and it has the following format
 - Word\tInteger\n
 - Output:
 - A textual file containing the content of the large file where the appearing words are substituted by the corresponding integers

Exercise #26 - Example

- Input files
 - Large textual file

```
TEST CONVERSION WORD TO INTEGER  
SECOND LINE TEST WORD TO INTEGER
```

- Small dictionary file

```
1      CONVERSION  
2      INTEGER  
3      LINE  
4      SECOND  
5      TEST  
6      TO  
7      WORD
```

Exercise #26 - Example

- Output file

```
5 1 7 6 2
```

```
4 3 5 7 6 2
```

Exercise #27

■ Categorization rules

■ Input:

- A large textual file containing a set of records
 - Each line contains the information about one single user
 - Each line has the format
 - UserId,Name,Surname,Gender,YearOfBirth,City,Education
- A small file with a set of business rules that are used to assign each user to a category
 - Each line contains a business rule with the format
 - Gender=<value> and YearOfBirth=<value> -> Category
 - Rules are mutually exclusive

Exercise #27

- Output:
 - One record for each user with the following format
 - The original information about the user plus the category assigned to the user by means of the business rules
 - Since the rules are mutually exclusive, there is only one rule applicable for each user
 - If no rules is applicable/satisfied by a user, assign the user to the “Unknown” category

Exercise #27 - Example

■ Users

User#1, John, Smith, M, 1934, New York, Bachelor
User#2, Paul, Jones, M, 1956, Dallas, College
User#3, Jenny, Smith, F, 1934, Philadelphia, Bachelor
User#4, Laura, White, F, 1926, New York, Doctorate

■ Business rules

Gender=M and YearOfBirth=1934 -> Category#1
Gender=M and YearOfBirth=1956 -> Category#3
Gender=F and YearOfBirth=1934 -> Category#2
Gender=F and YearOfBirth=1956 -> Category#3

Exercise #27 - Example

■ Output

User#1,John,Smith,M,1934,New York,Bachelor,Category#1

User#2,Paul,Jones,M,1956,Dallas,College,Category#3

User#3,Jenny,Smith,F,1934,Los Angeles,Bachelor,Category#2

User#4,Laura,White,F,1926,New York,Doctorate,Unknown

Exercise #28

- Mapping Question-Answer(s)
 - Input:
 - A large textual file containing a set of questions
 - Each line contains one question
 - Each line has the format
 - QuestionId,Timestamp,TextOfTheQuestion
 - A large textual file containing a set of answers
 - Each line contains one answer
 - Each line has the format
 - AnswerId,QuestionId,Timestamp,TextOfTheAnswer

Exercise #28

- Output:
 - One line for each pair (question,answer) with the following format
 - QuestionId,TextOfTheQuestion, AnswerId,TextOfTheAnswer

Exercise #28 - Example

■ Questions

Q1,2015-01-01,What is ..?

Q2,2015-01-03,Who invented ..

■ Answers

A1,Q1,2015-01-02,It is ..

A2,Q2,2015-01-03,John Smith

A3,Q1,2015-01-05,I think it is ..

Exercise #28 - Example

■ Output

Q1,What is ..?,A1,It is ..

Q1,What is ..?,A3,I think it is ..

Q2,Who invented ..,A2,John Smith

Exercise #29

■ User selection

■ Input:

- A large textual file containing a set of records
 - Each line contains the information about one single user
 - Each line has the format
 - UserId,Name,Surname,Gender,YearOfBirth,City,Education
- A large textual file with pairs (UserId, MovieGenre)
 - Each line contains pair UserId, MovieGenre with the format
 - UserId,MovieGenre
 - It means that UserId likes movies of genre MovieGenre

Exercise #29

- Output:
 - One record for each user that likes both Commedia and Adventure movies
 - Each output record contains only Gender and YearOfBirth of a selected user
 - Gender,YearOfBirth
 - Duplicate pairs must not be removed

Exercise #29 - Example

■ Users

User#1, John, Smith, M, 1934, New York, Bachelor
User#2, Paul, Jones, M, 1956, Dallas, College
User#3, Jenny, Smith, F, 1934, Philadelphia, Bachelor

■ Likes

User#1, Commedia
User#1, Adventure
User#1, Drama
User#2, Commedia
User#2, Crime
User#3, Commedia
User#3, Horror
User#3, Adventure

Exercise #29 - Example

- Output

M,1934

F,1934