

# BigData Spark Hands-On 2 [part 1]

## Spark read/write, Csv, Json

[arnaud.nauwynck@gmail.com](mailto:arnaud.nauwynck@gmail.com)

Esilv 2024

# Outline

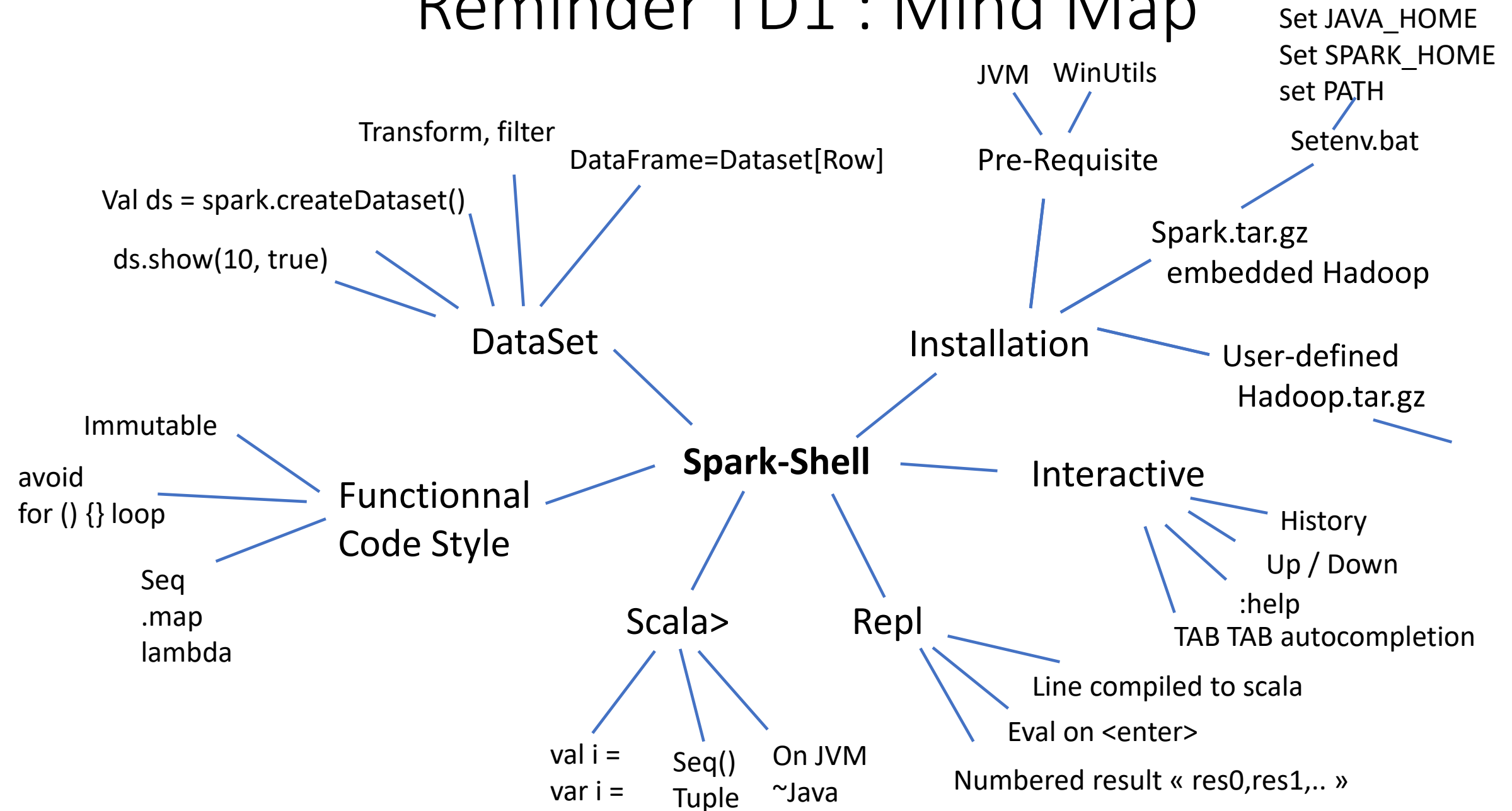
Reminder on TD1

Questions ? Problems on TD1 ?

Objective of TD2: mostly Files Input-Output

Pre-requisistes: same as TD1: « spark-shell »

# Reminder TD1 : Mind Map



# Objectives of Hands-On

1/ read CSV files, convert options

2/ read combine several files

3/ others file format: json, nd-json, xml, text

4/ write to PARQUET files dir  
repartitionning

5/ Discover PARQUET format optimisations

# Exercise 1: edit a csv file with few lines + Header

Edit a CSV file, with columns « id, text, boolean, date, doubleValue, .. »

Use either a text editor: Notepad++, SublimeText, IntelliJ / Eclipse, ...

First line should be « header » : columns names

All other lines = values, separated by « ; »

Strings may contain special ';' character, but should be wrapped with double quotes

Save as file « c:\data\sample-data1.csv »

# Exercise 1..

```
id;text;boolean;date;doubleValue
1;hello 1;true;2020/01/24;3.1415926
2;hello 2;false;2020/01/25;2.71828
3;hello 3;true;2020/01/26;1.4142135
4;"hello multi-lines\n line 2..\n line 3";false;2020/01/27;0.0
5;"hello line containing ;;; chars";false;2020/01/28;1.0
```

# Exercise 2 : Read CSV file from spark-shell

```
val sample1Ds = spark.read. ....
```

use

```
spark.read .option( ... ) .format(« csv ») .load ( path )
```

OR equivalent

```
spark.read .option( ... ) .csv( path )
```

CHECK :

1/ correct path

2/ option to set the field delimiter

3/ option to interpret the first header line correctly

4/ cast or interpret types correctly

5/ use `sample1Ds.show(20, false)` to check lines

6/ use `sample1Ds.printSchema` to check types

... string,boolean,int should be OK

for date .. Cf Exercise 3 or 5 !

## Exercise 3: use `.schema(« col1 type1, ..»)`

Add

```
spark.read .. .schema( « name1 type1, name2 type2..... »)
```


to force column types

(but maybe not date columns that may need explicit parsing .. cf next !)



# Exercise 3: Spark can not inferSchema for date ...

<https://stackoverflow.com/questions/66935214/spark-reading-csv-with-specified-date-format>

 **stackoverflow**


Products

Search...

Add a comment

Home

PUBLIC


 **Questions**

Tags

Users



Companies

COLLECTIVES

 Explore Collectives


TEAMS

**Stack Overflow for Teams** – Start collaborating and sharing organizational knowledge.


 


1 Answer


Sorted by: Highest score (default)

 Spark cannot infer `date` type. There are 2 possibilities:

2







```
val df = spark.read
    .option("header", true)
    .option("dateFormat", "yyyy.MM.dd")
    .schema("date date, something int")
    .csv("oo2.csv")
```

2. a workaround like:

```
val oo = spark.read.
    option("header", "true").
    //infer schema for other types
    option("inferSchema", "true").
    csv("oo2.csv").
    //manually create a new column with date
    withColumn("new_date", to_date(col("date"), "yyyy.MM.dd."))
```

# Exercise 4: use schema for file with no Header line

copy file « sample-data1.csv » without the header line  
save as « sample-data1-no-header.csv »

Read CSV file using option(« schema », « ..... »)

## Exercise 5: post processing conversion String->Date

From Dataset sample1Ds (from exercise3/)  
... with date as String

Define new column « parsed\_date » with typed date,  
then drop previous « date » text column,  
then rename column

# Exercise 6: real CSV Files, example of Open Data

The screenshot shows the data.gouv.fr website interface. At the top, there's a search bar with the text 'Recherche' and a magnifying glass icon. Below the search bar, there's a navigation menu with links: 'Données', 'Réutilisations', 'Organisations', 'Commencer sur data.gouv.fr', 'Actualités', and 'Nous contacter'. A blue banner below the navigation menu says 'Découvrez le nouveau score de qualité des métadonnées.' with links 'Donnez-nous votre avis.' and 'En savoir plus.'.

The main section is titled 'Jeux de données' and shows 'Rechercher parmi les 42 894 jeux de données sur data.gouv.fr'. A search bar contains the word 'adresse' and a 'Recherche' button. To the left, there are filters for 'Organisations' (set to 'Toutes les orga...') and 'Mots clés' (set to 'Tous les mots-c...').

The search results show '2234 résultats' and 'Trier par: Pertinence'. The first result is 'Base Adresse Nationale BAN' by 'Base Adresse Nationale'. It includes a description: 'La Base Adresse Nationale est l'une des neuf bases de données du service public des données de référence. Elle est la seule base de données d'adresses officiell...'. It also shows '41 réutilisations' and '50 favoris'.

Below the result, there's a section titled 'Fichiers (1)' with a sub-section 'FICHIER PRINCIPAL'. It contains a link 'Accéder à la documentation et aux données' and a note 'Mis à jour le 1 juillet 2022 — www — 985 téléchargements'.

Blue arrows indicate the flow from the search results to the 'Fichiers' section, then to the download icon, and finally to the download options panel.

<https://www.data.gouv.fr>

<https://www.data.gouv.fr/fr/datasets/base-adresse-nationale/>

The screenshot shows the 'Télécharger les données' panel. It has a section titled 'Format BAL' with the text 'Fichier CSV au format [BAL 1.3 \(AITE\)](#)'. Below this, there's a bullet point: '• Plusieurs positions par adresse'. At the bottom, there's a blue button with a download icon and the text 'Téléchargez au format BAL'.

**Base Adresse Nationale:**  
number, street, city, zipCode, latitude, longitude ...

1 CSV file 3.3 Go  
.. Or split by 990 files x 33 Ko

# Exercise 6 ... download + uncompress files

<https://adresse.data.gouv.fr/data/ban/adresses/latest/csv-bal/>

## Index of /data/ban/adresses/latest/csv-bal/

<a href="#">../</a>		
<a href="#">adresses-01.csv.gz</a>	26-Sep-2022 07:46	6788728
<a href="#">adresses-02.csv.gz</a>	26-Sep-2022 07:46	6264425
<a href="#">adresses-03.csv.gz</a>	26-Sep-2022 07:46	4671417
<a href="#">adresses-04.csv.gz</a>	26-Sep-2022 07:45	2596804
<a href="#">adresses-05.csv.gz</a>	26-Sep-2022 07:45	2303802
<a href="#">adresses-06.csv.gz</a>	26-Sep-2022 07:46	5746570
<a href="#">adresses-07.csv.gz</a>	26-Sep-2022 07:46	5274174
<a href="#">adresses-08.csv.gz</a>	26-Sep-2022 07:46	3186529
<a href="#">adresses-09.csv.gz</a>	26-Sep-2022 07:46	3306036
<a href="#">adresses-10.csv.gz</a>	26-Sep-2022 07:36	3723847
<a href="#">adresses-11.csv.gz</a>	26-Sep-2022 07:36	6426680
<a href="#">adresses-12.csv.gz</a>	26-Sep-2022 07:36	3896309
<a href="#">adresses-13.csv.gz</a>	26-Sep-2022 07:38	13520343
...	...	...
<a href="#">adresses-977.csv.gz</a>	26-Sep-2022 07:44	20
<a href="#">adresses-978.csv.gz</a>	26-Sep-2022 07:44	20
<a href="#">adresses-984.csv.gz</a>	26-Sep-2022 07:44	20
<a href="#">adresses-986.csv.gz</a>	26-Sep-2022 07:44	20
<a href="#">adresses-987.csv.gz</a>	26-Sep-2022 07:44	80482
<a href="#">adresses-988.csv.gz</a>	26-Sep-2022 07:45	734636
<a href="#">adresses-989.csv.gz</a>	26-Sep-2022 07:44	20
<a href="#">adresses-france.csv.gz</a>	26-Sep-2022 07:51	676109377

} Exercise: Download few files  
01, 02, ... 04

} 990 x small csv.gz files

} ALL in ONE = 1 x Big csv.gz files

# Exercise 6: load individual CSV files, then do union of DataSets

a/ load files:

address-01.csv -> in dataset01

address-02.csv -> in dataset02

..

address-03 .csv -> in dataset03

b/ do union of datasets: dataset01.**union**(dataset02) .. **union**(dataset04)

c/ check that count of union = sum of count of individual files

# Exercise 7: load multiple CSV files from 1 directory

- a/ create an empty directory, for example « C:\data\OpenData-gouv.fr\bal\bal-split-files »
- b/ put your few address files in it (adress-01.csv, -02, ..-03, -04 )  
ensure you have NO other csv files in it
- c/ read all files using 1 command: `spark.read ...` taking path of directory (instead of path of file)
- d/ check that count is similar to exercise 7/

# Exercise 8 ... repeat query before/after .cache() explain which query a/b/c/.. perform IO reads

scala> spark.read. ... « infer »true	←	a/ Read? ... for schema/statistics, but data not retained in memory
scala> allAdressCsvDs.count	←	b/ Read ?
scala> allAdressCsvDs.count	←	c/ Read ?
scala> allAdressCsvDs. <b>cache()</b>	←	d/ Read ?
scala> allAdressCsvDs.count	←	e/ Read ?
scala> allAdressCsvDs.count	←	f/ Read ?



# Objectives of Hands-On



1/ read CSV files, convert options



2/ read combine several files

3/ others file format: json, nd-json, xml, text

4/ write to PARQUET files dir  
repartitionning

5/ Discover PARQUET format optimisations

10 mn pause

## APPENDIX ...

If you have finished exercises 1-8, but your friends are still struggling ?

Idea 1 / help your struggling friends

Idea 2/ imagine processing queries on OpenData addresses  
( see next )

# APPENDIX 1: Exercise 6 was loading « addresses » ...

## **What could you process out of this ?**

Suggestions:

1. Extract all city names with zipCode, average latitude/longitude
2. Extract all distinct street names
3. Extract number of addresses and number of streets per city
4. Detect all street named to a person « firstname Lastname » given a list of known firstname
5. Count how many times « Victor Hugo » is present
6. Count cities having neither « Victor Hugo », nor « Balzac », or « Picasso »
7. Detect cities having one of « Leningrad », « Stalingrad », « Maurice Thorez »
8. ...

# Objectives of Hands-On



1/ read CSV files, convert options



2/ read combine several files



3/ others file format: json, nd-json, xml, text

4/ write to PARQUET files dir  
repartitionning

5/ Discover PARQUET optimisations

6/ Discover Spark UI

# Exercise 9: save as json, load json file

a/ write addresses Dataset (from CSV) to JSON

b/ look at written files

is it a valid « .json » file, containing 1 array of objects ?

Or ND-json = New-lines Delimited Json

compare file size of csv and nd-json

c/ load dataset from ND-json file

# Exercise 10 : analyze written json files

**Repeat the same operation twice, and notice that**

1/ you can not specify the file names .. Only an empty directory name !

2/ the file names are generated with a unique UUID  
part-00{partNumber}-{UUID}-c000.json

3/ during write time... spark write to « \_temporary » sub dir, then rename at the end

4/ spark has written several files ...

If using the full dataset of 3Giga .. Spark write 26 json files

# (Optional) Exercise 11 : convert ND-json to valid JSON array File

## ... then load from spark

To convert ... you need to

1. add a START\_ARRAY delimiter at line 1 : char « [«
2. finish all lines with char « , » ... except the last line !!
3. finish END\_ARRAY at last line : char « ] »

a/ edit manually a little file (~10 lines) / use regexp / or write a program(?)

b/ load this real json « multiline » file from spark



# Exercise 12 : Read plain « Text » lines from file

a/ edit a file containing many lines of text

b/ load this text file from spark

c/ count number of lines

d/ what is the difference between `spark.read.textFile()` and `spark.read.text()` ?

# Exercise 13 : What about xml format ?

## Questions

a/ do you see a « xml() » method in spark.read. ?

b/ why ?

c/ suppose you want to extract known <element>..</element> from a <root>..</root>  
... How would you convert to a Dataset ?

# Exercise 13 ... Example of « Big » Xml dump files

<https://dumps.wikimedia.org/enwiki/20220901/>

## More than 100 Gigas of Xml files

2022-09-02 04:32:53   **done**   Articles, templates, media/file descriptions, and primary meta-pages, in multiple bz2 streams, 100 pages per stream

[enwiki-20220901-pages-articles-multistream1.xml-p1p41242.bz2](#) 251.2 MB

[enwiki-20220901-pages-articles-multistream-index1.txt-p1p41242.bz2](#) 221 KB

[enwiki-20220901-pages-articles-multistream2.xml-p41243p151573.bz2](#) 336.5 MB

[enwiki-20220901-pages-articles-multistream-index2.txt-p41243p151573.bz2](#) 638 KB

[enwiki-20220901-pages-articles-multistream3.xml-p151574p311329.bz2](#) 364.6 MB

[enwiki-20220901-pages-articles-multistream-index3.txt-p151574p311329.bz2](#) 820 KB

[enwiki-20220901-pages-articles-multistream4.xml-p311330p558391.bz2](#) 406.4 MB

[enwiki-20220901-pages-articles-multistream-index4.txt-p311330p558391.bz2](#) 1.3 MB

[enwiki-20220901-pages-articles-multistream5.xml-p558392p958045.bz2](#) 436.6 MB

# Objectives of Hands-On



1/ read CSV files, convert options



2/ read combine several files



3/ others file format: json, nd-json, xml, text

4/ write to PARQUET files dir  
repartitionning

5/ Discover PARQUET format optimisations

## APPENDIX ...

If you have finished exercises 9-13, but your friends are still struggling ?

Idea 1 / help your struggling friends

Idea 2/ bonus exercise 14 on textFile

# Exercise 14 : the « World Count »

## Hello-World standard program of BigData

From dataset of text lines from exercise 12/

split all lines into words

... For this, start by « map() » replacing all punctuation marks by a single space

... Then replace 2 successive spaces by a single space

... Then split by space

... then use « flatMap() » instead of « map() »

Finally extract words count

10 mn pause

Continue...

Please open next document: Part2  
TD2-Part2 (PARQUET)