



Amina Dahmouni

Classe : II-BDCC2

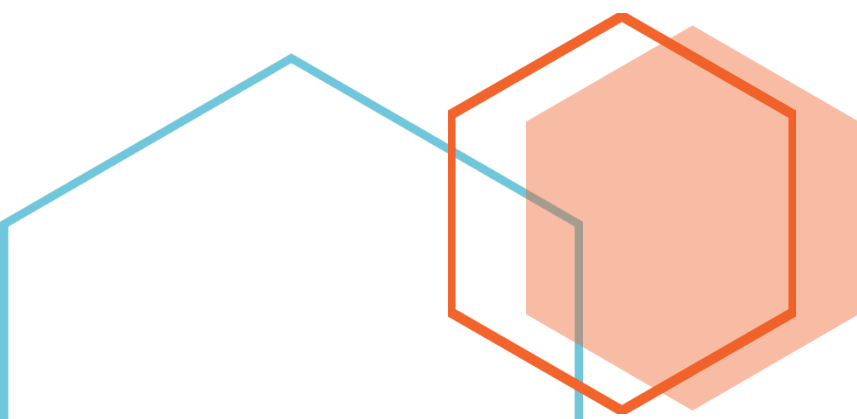
# [Compte Rendu]

---

## Tp 8 : Spark SQL

*Big Data: Fondements et Architectures de stockage*

L'Ecole Normale Supérieure de l'Enseignement Technique de  
Mohammedia (ENSET)



## 1. DataFrame :

### a) Employees.Json

```
1 [{"id": 1,"name": "Amina","phone": "0620202020","salary": 13000.00,"age": 22,"departement": "Informatique"},
2 {"id": 2,"name": "Bouchra","phone": "0620202020","salary": 23000.00,"age": 32,"departement": "Mécanique"},
3 {"id": 3,"name": "Yousra","phone": "0620202020","salary": 12500.00,"age": 45,"departement": "Informatique"},
4 {"id": 4,"name": "Zakiya","phone": "0620202020","salary": 24000.00,"age": 42,"departement": "Informatique"},
5 {"id": 5,"name": "Souad","phone": "0620202020","salary": 13000.00,"age": 22,"departement": "Mécanique"}]
```

### Employee.java

```
1 package ma.enset.Employee;
2
3 import java.io.Serializable;
4 public class Employee implements Serializable {
5     private Long id;
6     private String name;
7     private String phone;
8     private double salary;
9     //Changer le type de l'age du int vers Long cas Application 2
10    private int age;
11    private String departement;
```

- Afficher les employées ayant un âge entre 30 et 35 :

```
Application1Json.java x employees.json x Employee.java x
1 package ma.enset.Employee.DataFrame;
2
3 import ...
7 public class Application1Json{
8     public static void main(String[] args) {
9         SparkSession ss=SparkSession.builder().appName("TP Spark SQL").
10             master("local[*]").getOrCreate();
11         Dataset<Row> df=ss.read().json( path: "employees.json");
12         //df.printSchema();
13         //df.show();
14         df.createOrReplaceTempView( viewName: "employees");
15         df.filter(col( colName: "age").between( lowerBound: 30, upperBound: 35)).show();
```

```
+-----+-----+-----+-----+-----+-----+
|_corrupt_record|age|departement| id|   name|   phone| salary|
+-----+-----+-----+-----+-----+-----+
|           null| 32|   Mécanique|  2|Bouchra|0620202020|23000.0|
+-----+-----+-----+-----+-----+-----+
```

- Afficher la moyenne des salaires de chaque département :

```
df.groupBy( col1: "departement").avg( ...colNames: "salary").show();
```

```
+-----+-----+
| departement|avg(salary)|
+-----+-----+
|Informatique|   18250.0|
|           null|         null|
|   Mécanique|   18000.0|
+-----+-----+
```

- Afficher le nombre de salariés par département :

```
df.groupBy( col1: "departement").count().show();
```

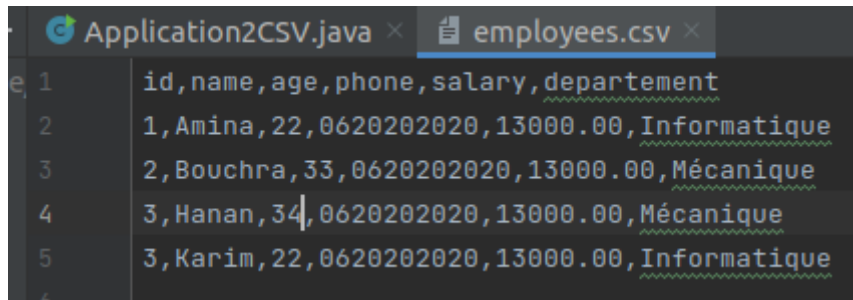
```
+-----+-----+
| departement|count|
+-----+-----+
|Informatique|    2|
|           null|    1|
|   Mécanique|    2|
+-----+-----+
```

- Afficher le salaire maximum de toutes les départements :

```
df.select(max( columnName: "salary")).show();
```

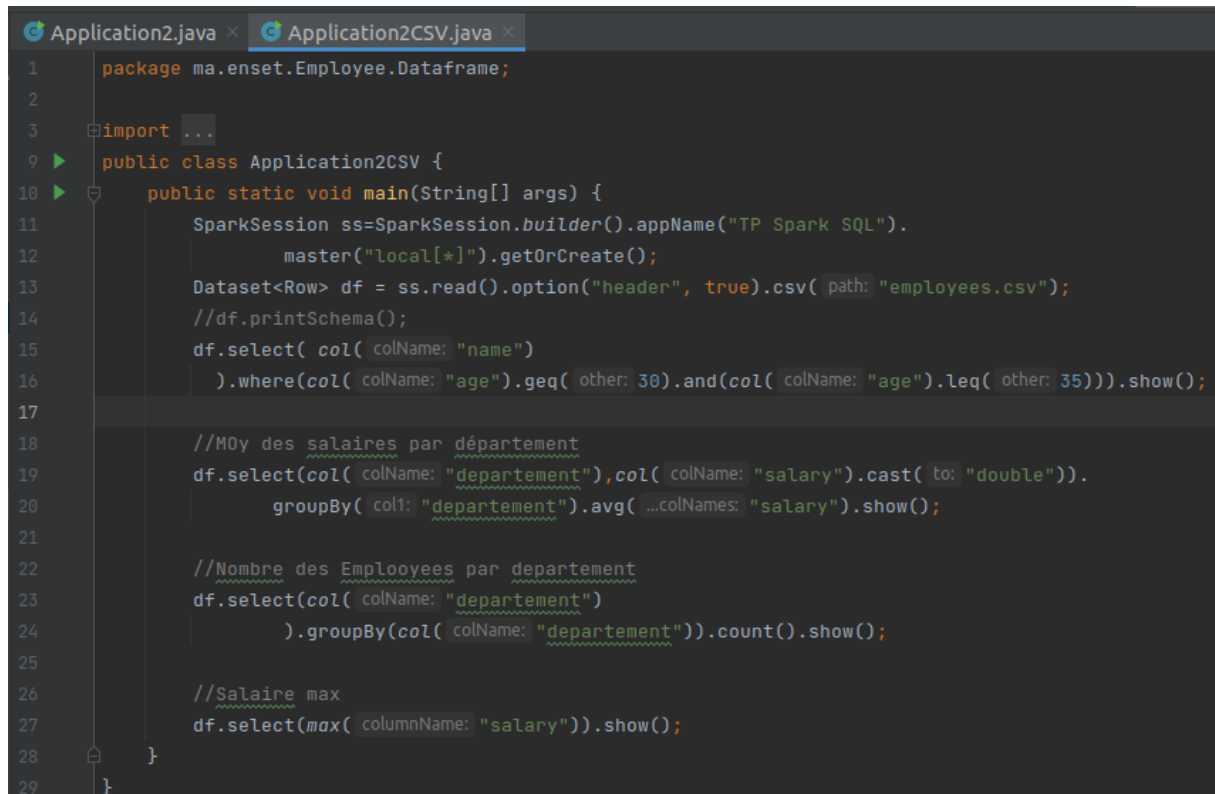
```
+-----+
|max(salary)|
+-----+
|    24000.0|
+-----+
```

## b) Employees.csv



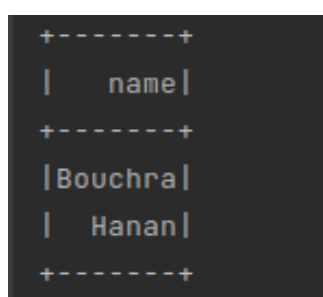
A screenshot of a text editor showing the content of a file named 'employees.csv'. The file contains five lines of CSV data. The first line is the header: 'id,name,age,phone,salary,departement'. The following four lines contain employee records: (1, Amina, 22, 0620202020, 13000.00, Informatique), (2, Bouchra, 33, 0620202020, 13000.00, Mécanique), (3, Hanan, 34, 0620202020, 13000.00, Mécanique), and (3, Karim, 22, 0620202020, 13000.00, Informatique).

id	name	age	phone	salary	departement
1	Amina	22	0620202020	13000.00	Informatique
2	Bouchra	33	0620202020	13000.00	Mécanique
3	Hanan	34	0620202020	13000.00	Mécanique
3	Karim	22	0620202020	13000.00	Informatique



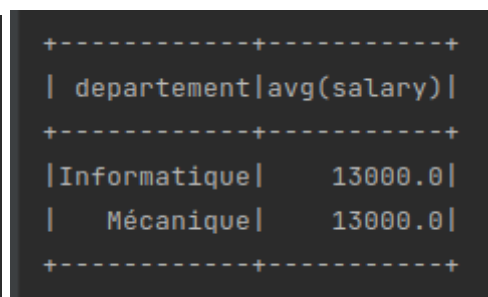
A screenshot of an IDE showing the code for 'Application2CSV.java'. The code imports the necessary Spark classes and defines a main method. It reads the 'employees.csv' file into a DataFrame, filters for employees aged 30 and above, and then performs three queries: calculating the average salary by department, counting the number of employees by department, and finding the maximum salary.

```
1 package ma.enset.Employee.Dataframe;
2
3 import ...
9 public class Application2CSV {
10     public static void main(String[] args) {
11         SparkSession ss=SparkSession.builder().appName("TP Spark SQL").
12             master("local[*]").getOrCreate();
13         Dataset<Row> df = ss.read().option("header", true).csv(path: "employees.csv");
14         //df.printSchema();
15         df.select(col(colName: "name")
16             ).where(col(colName: "age").geq( other: 30).and(col(colName: "age").leq( other: 35))).show();
17
18         //Moy des salaires par département
19         df.select(col(colName: "departement"),col(colName: "salary").cast(to: "double")).
20             groupBy(col: "departement").avg(...colNames: "salary").show();
21
22         //Nombre des Emplooyees par departement
23         df.select(col(colName: "departement")
24             ).groupBy(col(colName: "departement")).count().show();
25
26         //Salaine max
27         df.select(max(columnName: "salary")).show();
28     }
29 }
```



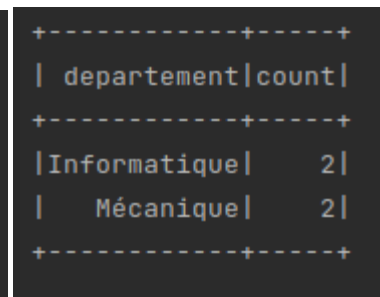
Terminal output showing the names of employees aged 30 and above. The output is formatted as a table with a header row and two data rows.

name
Bouchra
Hanan



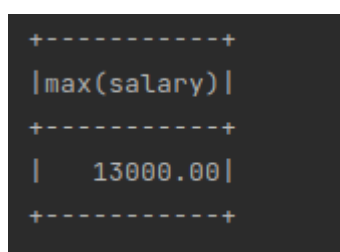
Terminal output showing the average salary by department. The output is formatted as a table with two columns: 'departement' and 'avg(salary)'.

departement	avg(salary)
Informatique	13000.0
Mécanique	13000.0



Terminal output showing the count of employees by department. The output is formatted as a table with two columns: 'departement' and 'count'.

departement	count
Informatique	2
Mécanique	2



Terminal output showing the maximum salary. The output is formatted as a table with one column: 'max(salary)'.

max(salary)
13000.00

## 2. DataSet :

Premièrement on modifie le type de l'attribut *age* par *Long* :

```
//changer le type de l'age  
2 usages  
private Long age;  
2 usages
```

### a) Employees.Json

```
Application1JsonDS.java  
1 package ma.enset.Employee.Dataset;  
2 import ...  
7 public class Application1JsonDS {  
8     public static void main(String[] args) {  
9         SparkSession ss=SparkSession.builder().  
10             appName("TP Spark SQL").  
11             master("local[*]").getOrCreate();  
12  
13         Encoder<Employee> employeeEncoder= Encoders.bean(Employee.class);  
14         Dataset<Employee> ds=ss.read().option("multiline",true).json( path: "employees.json").as(employeeEncoder);  
15         //ds.printSchema();  
16         System.out.println("Question 1");  
17         ds.filter((FilterFunction<Employee>) emp->emp.getAge()>=30&&emp.getAge()<=35).show();  
18  
19  
20         System.out.println("Question 2");  
21         ds.select(col( colName: "departement"), col( colName: "salary")  
22             ).groupBy(col( colName: "departement")).mean( ...colNames: "salary").show();  
23  
24  
25         System.out.println("Question 3");  
26         ds.select(col( colName: "departement"))  
27             .groupBy(col( colName: "departement")).count().show();  
28  
29  
30         System.out.println("Question 4");  
31         ds.select(max( columnName: "salary")).show();  
32  
33     }  
34 }
```

## b) Employees.csv

```
Application2CSVDS.java x
11 ▶ public class Application2CSVDS {
12 ▶     public static void main(String[] args) {
13         SparkSession ss=SparkSession.builder().
14             appName("TP Spark SQL").
15             master("local[*]").getOrCreate();
16
17         Encoder<Employee> employeeEncoder= Encoders.bean(Employee.class);
18         Dataset<Employee> ds = ss.read().format("csv").option("delimiter",",").option("header", true)
19             .option("inferSchema", "true").option("charset", "UTF8")
20             .csv(path: "employees.csv").as(employeeEncoder);
21
22         System.out.println("Question 1");
23         System.out.println("csv file");
24         ds.filter((FilterFunction<Employee>) emp->emp.getAge()>=30&&emp.getAge()<=35).show();
25
26         System.out.println("Question 2");
27         ds.select(col(colName: "departement"), col(colName: "salary")
28             ).groupBy(col(colName: "departement")).mean(...colNames: "salary").show();
29
30         System.out.println("Question 3");
31         ds.select(col(colName: "departement"))
32             .groupBy(col(colName: "departement")).count().show();
33
34         System.out.println("Question 4");
35         ds.select(max(columnName: "salary")).show();
36
37     }
38 }
39 }
```