

Traitements Big Data - Apache Spark : Spark SQL Et Dataframes

Spark  SQL



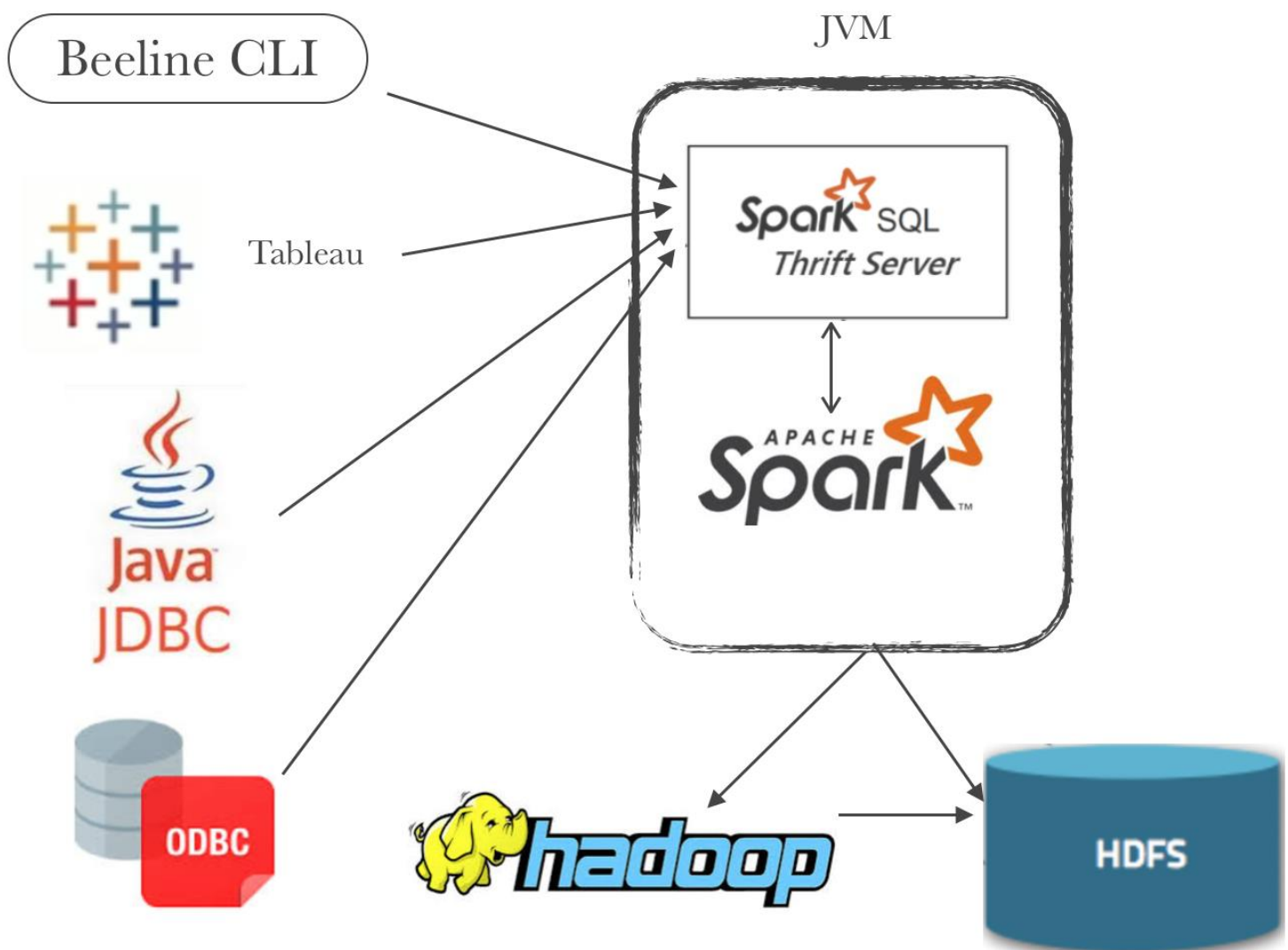
Réalisé par :
Mouad Riali
Kamal Addi

Encadré par :
Pr D.Zaidouni

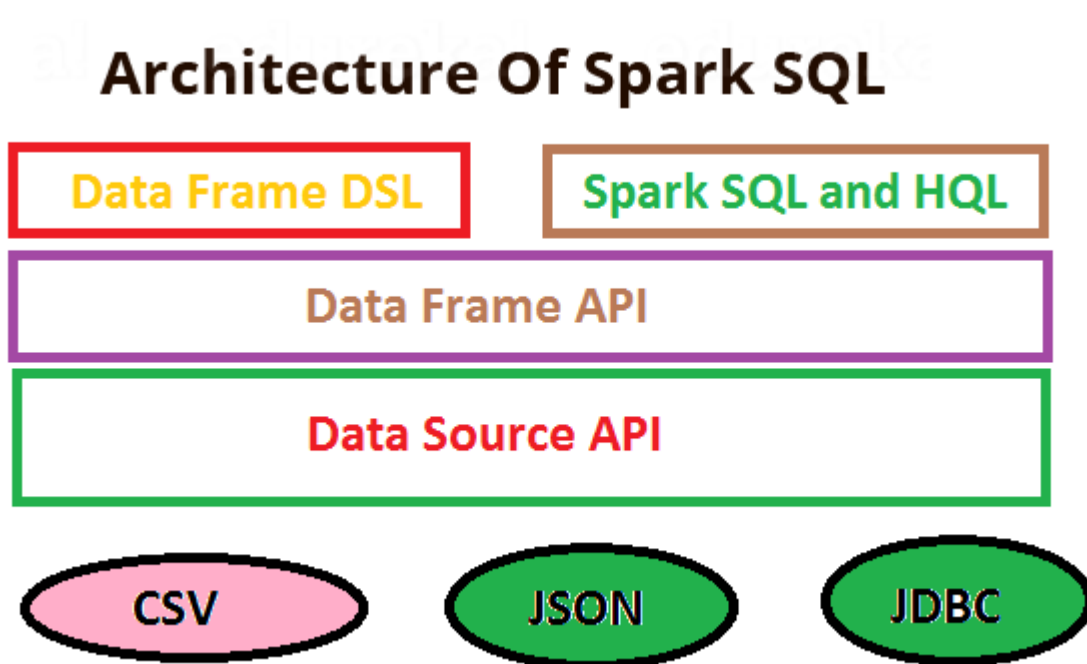
I- Spark SQL :

Spark SQL, est un composant du framework Apache Spark, est utilisé pour effectuer des traitements sur des données structurées en exécutant des requêtes de type SQL sur les données Spark. Il nous permet d'exécuter des requêtes SQL, en utilisant les outils BI et de visualisation traditionnels, après une étape d'ETL sur des données stockées sous différents formats, comme CSV, JSON ou Parquet, ou des données stockées dans des bases de données. Spark SQL permet d'exposer les jeux de données Spark via API JDBC, ODBC...

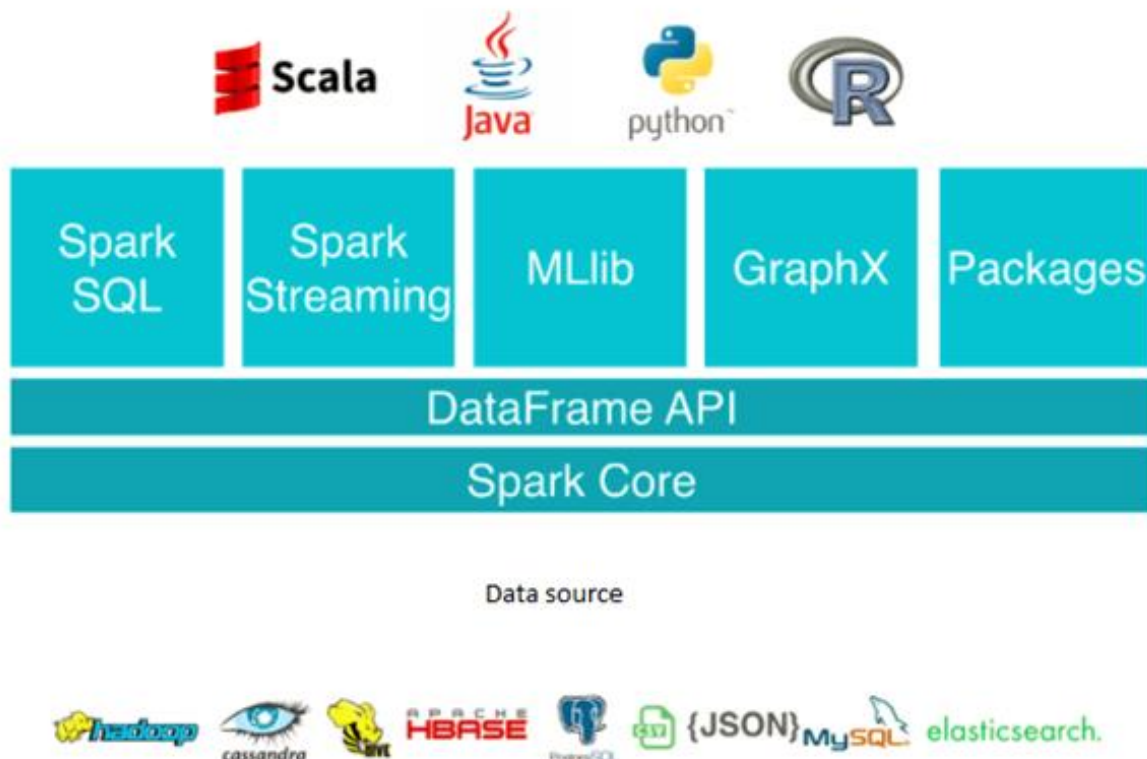
Les données provenant de plusieurs sources peuvent être transférées dans Spark, puis exposées sous forme de tables, ces tables sont ensuite rendues accessibles en tant que source de données JDBC / ODBC via Spark SQL. Nous avons plusieurs clients tels que Beeline CLI, JDBC, ODBC, des outils BI comme Tableau...etc. disponibles pour se connecter au serveur et visualiser nos données. L'image suivante illustre ce que je viens d'expliquer :



1- Architecture de Spark SQL :



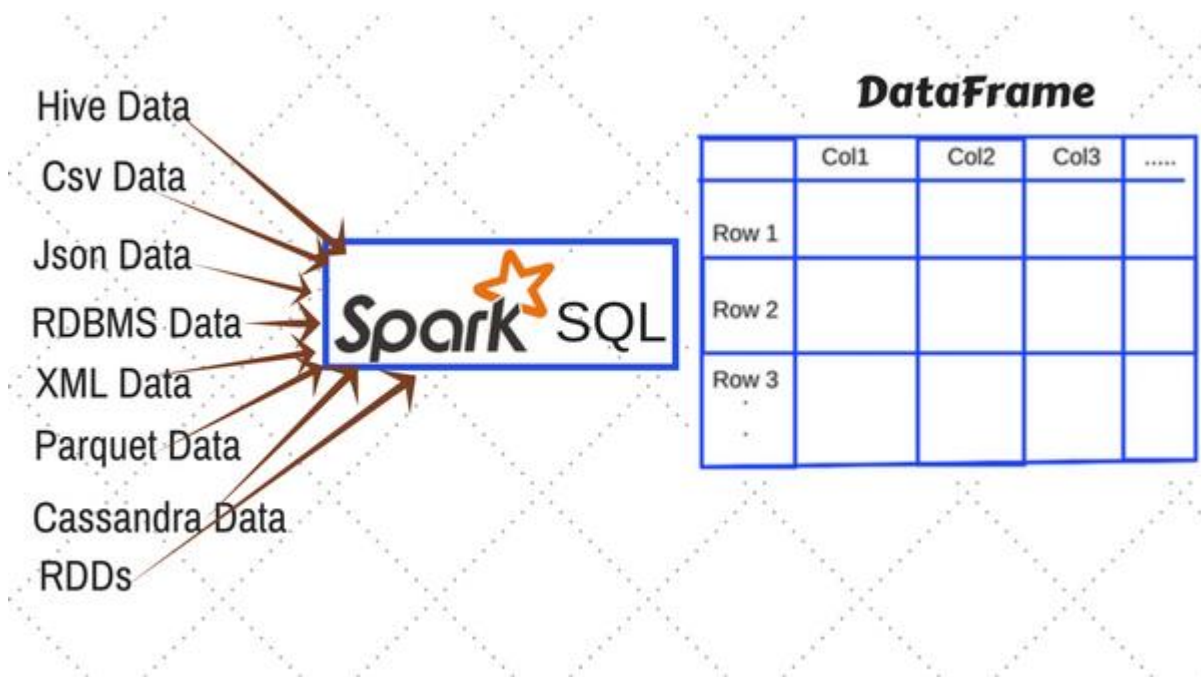
Spark SQL est le composant qui vient au dessus de la couche Core et qui introduit une nouvelle abstraction de données structurées et semi-structurées. Il fournit des fonctions qui se basent sur Spark Core et ses APIs Scala, Java, Python, SQL et R.



Spark SQL apporte une vision semi-structurée, les DataFrames, ce qui permet d'extraire, transformer et de charger des données sous différents formats (csv, Json, Parquet, base de données).

Un DataFrame est une collection de données distribuées, organisées en colonnes nommées. Ce concept est basé sur celui des data frames du langage R et est similaire à une table dans le monde des bases de données relationnelles. Notez que c'est en fait ce qu'on appelait SchemaRDD dans les précédentes versions de l'API Spark SQL qui a été renommé DataFrame.

Création d'un Dataframe en Spark à l'aide de Spark SQL :

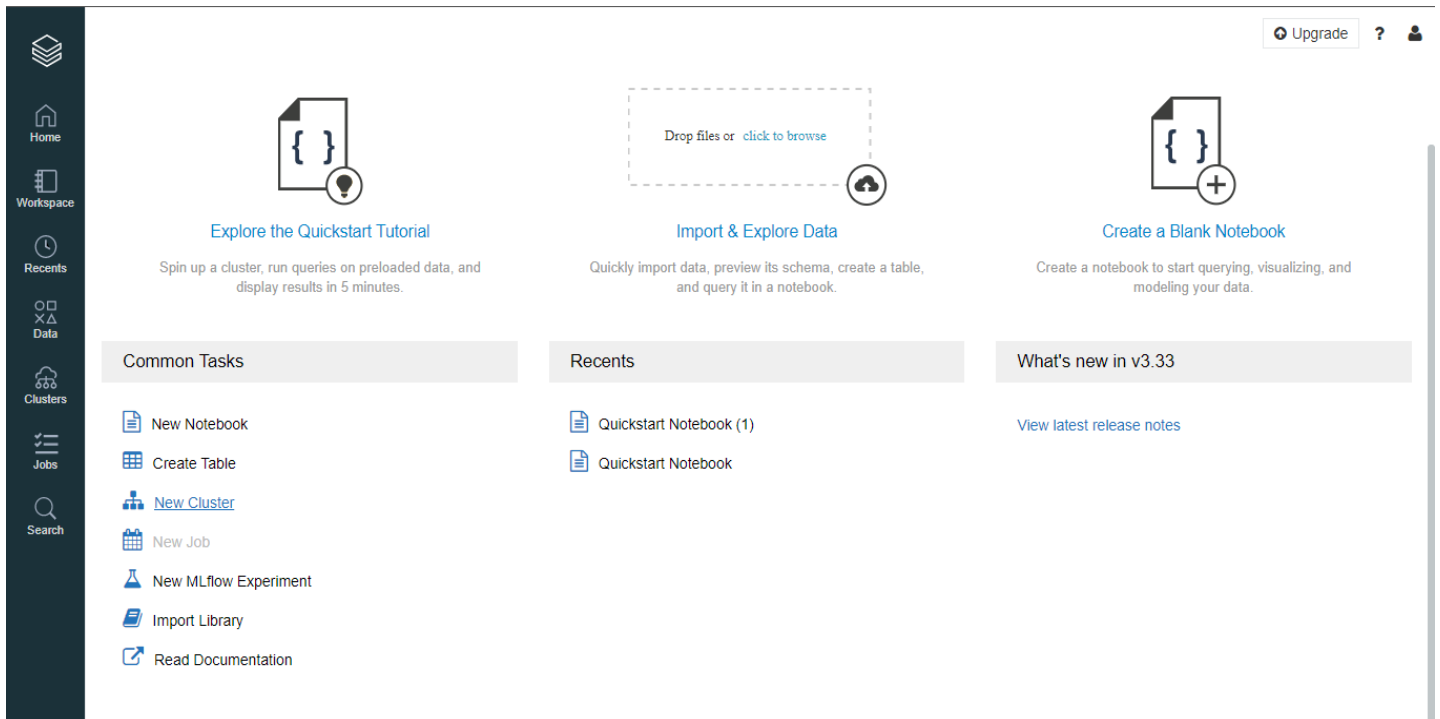


2- Exemple d'application Spark SQL dans le Cloud Databricks :

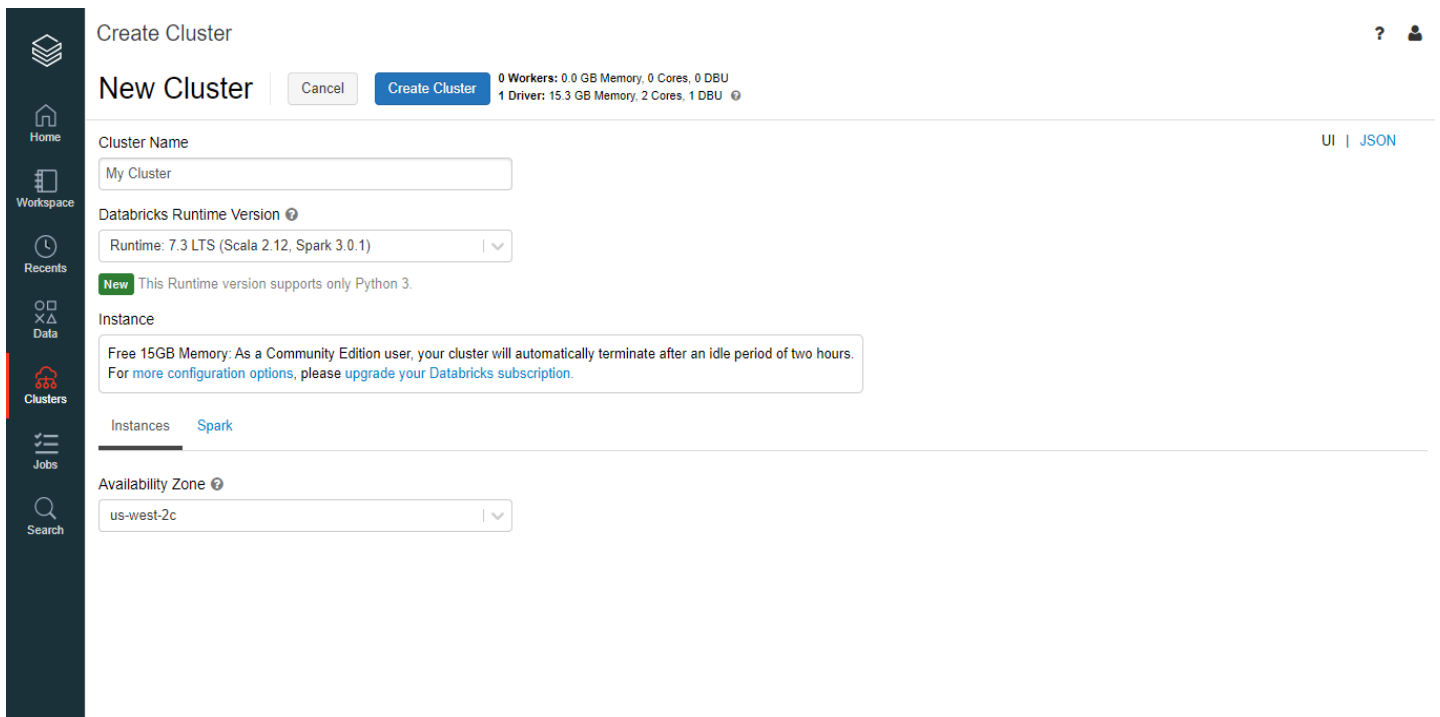
- Créer un cluster :

Avec Cloud Databricks on peut créer un cluster rapidement et facilement en quelques cliques.

Dans la barre "Common tasks", cliquons sur "New Cluster" :



Dans la fenêtre qui s'ouvre on tape le nom du cluster "My Cluster" dans notre cas, puis on choisit la version de Spark qu'on veut utiliser, ici la version 3.0.1, enfin on clique "Create Cluster" en bleu :



Dans l'onglet "Event Log" on peut visualiser les différents événements avec leurs date et heure de création :

The screenshot shows the Databricks Clusters management interface. The left sidebar contains navigation icons for Home, Workspace, Recents, Data, Clusters (highlighted), Jobs, and Search. The main header shows 'Clusters /' and a user profile icon. Below the header, there's a section for 'My Cluster' with buttons for Edit, Clone, Restart, Terminate, and Delete. A navigation bar includes links for Configuration, Notebooks, Libraries, Event Log (selected), Spark UI, Driver Logs, Metrics, Apps, and Spark Cluster UI - Master. The Event Log table has columns for Event Type, Time, and Message. It contains three entries: DRIVER_HEALTHY, RUNNING, and CREATING. A 'Filter by Event Type...' dropdown and a 'Refresh' button are at the top right of the table.

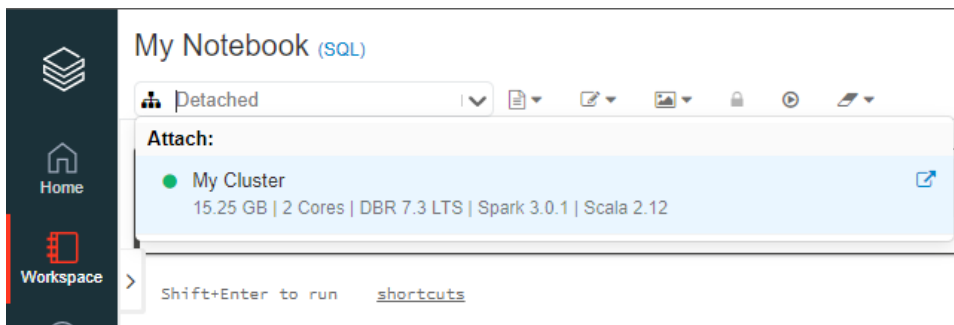
Event Type	Time	Message
DRIVER_HEALTHY	2020-11-24 22:55:29 +01	Driver is healthy.
RUNNING	2020-11-24 22:54:29 +01	Cluster is running.
CREATING	2020-11-24 22:52:01 +01	Cluster creation requested by kamal-addi@outlook.fr.

- Attacher le notebook au cluster :

Nous allons tout d'abord créer un notebook, de même on choisit l'onglet Create Notebook, on choisit comme nom My Notebook, default language SQL, et enfin notre Cluster que nous avons déjà créer :

The left screenshot shows the 'Create Notebook' dialog with 'Name' set to 'My Notebook', 'Default Language' set to 'SQL', and the 'Cluster' dropdown menu open, showing options: Python, Scala, SQL (highlighted), and R. The right screenshot shows the same dialog with 'Cluster' set to 'My Cluster'. A tooltip for 'My Cluster' is visible, showing details: 15.25 GB | 2 Cores | DBR 7.3 LTS | Spark 3.0.1 | Scala 2.12. Both dialogs have 'Cancel' and 'Create' buttons.

Attachons maintenant le notebook à notre cluster, dans la barre de menu du Notebook, sélectionnez "Detached" puis "My Cluster".



Sur ce notebook on peut créer des tables faire des requêtes SQL, faire des manipulations etc... :

Cmd 2

Les commandes suivantes crée une table à partir d'un ensemble de données Databricks

Cmd 3

```
1 DROP TABLE IF EXISTS diamonds;
2
3 CREATE TABLE diamonds
4 USING csv
5 OPTIONS (path "/databricks-datasets/Rdatasets/data-001/csv/ggplot2/diamonds.csv", header "true")
```

► (1) Spark Jobs
OK

Command took 7.79 seconds -- by kamal-addi@outlook.fr at 11/24/2020, 11:41:24 PM on My Cluster

Afficher la totalité de la table créer :

Cmd 5

```
1 SELECT * from diamonds
```

► (1) Spark Jobs

	_c0	carat	cut	color	clarity	depth	table	price	x	y	z
1	1	0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
2	2	0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
3	3	0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
4	4	0.29	Premium	I	VS2	62.4	58	334	4.2	4.23	2.63

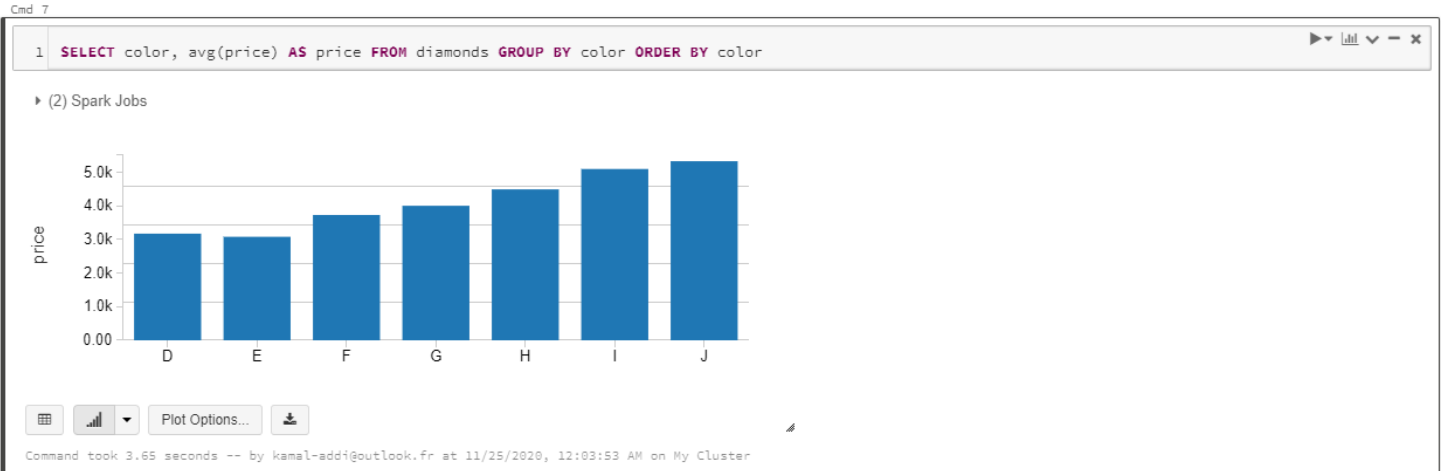
Showing the first 1000 rows.

Command took 1.92 seconds -- by kamal-addi@outlook.fr at 11/24/2020, 11:42:36 PM on My Cluster

La requête suivante manipule les données et affiche les résultats :

Plus précisément, la commande:

1. Sélectionne les colonnes "color" et "price" et la moyenne du prix et regroupe les résultats par couleur et les affiche par ordre.
2. Affiche un tableau des résultats.



Changer le format graphique en tableau :

Sous l'histogramme, on peut choisir le format graphique qui répond à nos besoins

Cmd 8

```
1 SELECT color, avg(price) AS price FROM diamonds GROUP BY color ORDER BY color
```

► (2) Spark Jobs

	color	price
1	D	3169.9540959409596
2	E	3076.7524752475247
3	F	3724.886396981765
4	G	3999.135671271697
5	H	4486.669195568401
6	I	5091.874953891553
7	J	5323.81801994302

Plot Options...

- Bar
- Scatter
- Map
- Line
- Area
- Pie
- Quantile
- Histogram
- Box plot
- Q-Q plot
- Pivot**
- Legacy charts

Changer le format graphique

Sous l'histogramme, on peut choisir le format graphique qui répond à nos besoins

• Création d'un dataframe :

En utilisant python on peut facilement créer des dataframes en une seule ligne :

```
diamonds = spark.read.csv("/databricks-datasets/Rdatasets/data-001/csv/ggplot2/diamonds.csv",  
header="true", inferSchema="true")
```

My Notebook (SQL)

La commande suivante créer un DataFrame à partir d'un ensemble de données Databricks :

```
1 %python  
2 diamonds = spark.read.csv("/databricks-datasets/Rdatasets/data-001/csv/ggplot2/diamonds.csv", header="true", inferSchema="true")
```

▶ (2) Spark Jobs

▼ diamonds: pyspark.sql.dataframe.DataFrame

```
_c0: integer  
carat: double  
cut: string  
color: string  
clarity: string  
depth: double  
table: double  
price: integer  
x: double  
y: double  
z: double
```

Command took 4.47 seconds -- by kamal-addi@outlook.fr at 11/25/2020, 11:09:24 AM on My Cluster

Et maintenant on peut faire des manipulations sur notre Dataframe **diamonds** :

La commande **display()** permet d'afficher les résultats et **head()** affiche les 5 premiers lignes du Dataframe :

Afficher les 5 premiers lignes de notre Dataframe :

```
1 %python  
2 display(diamonds.head(5))
```

▶ (4) Spark Jobs

	_c0	carat	cut	color	clarity	depth	table	price	x	y	z
1	1	0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
2	2	0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
3	3	0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
4	4	0.29	Premium	I	VS2	62.4	58	334	4.2	4.23	2.63
5	5	0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75

Showing all 5 rows.

Command took 1.06 seconds -- by kamal-addi@outlook.fr at 11/25/2020, 11:16:57 AM on My Cluster

On peut faire appel à la même requête qu'on a fait précédemment avec SQL, mais cette fois avec les fonctions python appliqués à notre Dataframe :





La table des "color", moyennes "price" qu'on a déjà fait avec SQL :

Cmd 16

```
1 %python
2 from pyspark.sql.functions import avg
3
4 display(diamonds.select("color","price").groupBy("color").agg(avg("price")).sort("color"))
```

► (2) Spark Jobs

	color	avg(price)
1	D	3169.9540959409596
2	E	3076.7524752475247
3	F	3724.886396981765
4	G	3999.135671271697
5	H	4486.669195568401
6	I	5091.874953891553
7	J	5323.81801994302

   Plot Options... 

Command took 2.73 seconds -- by kamal-addi@outlook.fr at 11/25/2020, 11:31:19 AM on My Cluster

Pour voir la totalité du Notebook consulter les liens suivants :

- [Big-Data/My Notebook.ipynb at main · addi-kamal/Big-Data \(github.com\)](#)
- [Big-Data/My Notebook.py at main · addi-kamal/Big-Data \(github.com\)](#)
- [Big-Data/My Notebook.sql at main · addi-kamal/Big-Data \(github.com\)](#)