Spark Cheat Sheet

SparkSQL - Transformations

`df = sqlContext.createDataFrame(data)`	Créer un DataFrame à partir d'une collection Python (liste)
`df = sqlContext.createDataFrame(data, ['name', 'age'])`	Créer un DataFrame à partir d'une collection Python (liste)
`df = sqlContext.read.text("file.txt")`	Crér un DataFrame à partir d'un fichier
`ageCol = people.age`	Créer un DataFrame à partie d'une colonne
`df.select('*')`	Sélectionner toutes les colonnes
`df.select('name', 'age')`	Sélectionne 1 ou plusieurs colonnes
`df.select(df.name, (df.age + 10).alias('age'))`	Sélectionner 2 colonnes, changer la valeur d'une colonne et renommer la colonne
`df.dropcolumn()`	Suppression d'une colonne.
`df.drop(df.age)`	Retourne un nouveau DataFrame
`lambda a, b : a + b`	Fontion anonyme. 1 expression
`slen = udf(lambda s: len(s), IntegerType())`	Fonction lambda ou nommée et type du retour
`innerJoinDF.filter(innerJoinDF.comment)`	Retourne un DataFrame dont les lignes respectent la/les condition(s)
`where(function)`	Retourne un DataFrame dont les lignes respectent la/les condition(s)
`df.distinct()`	Retourne un DataFrame avec les lignes uniques
`orderBy(cols, *kw)`	Retourne un DataFrame dans l'ordre croissant ou décroissant
`df.sort("age", ascending = False)`	Retourne un DataFrame dans l'ordre croissant ou décroissant
`df.select(explode(df.intlist).alias('anInt'))`	Chaque élément est dans une nouvelle ligne

SparkSQL - Actions

`df.show(n, truncate)`	Affiche les n premières lignes du DataFrame
`df.take(n)`	Affiche les n premières lignes sous forme de une liste
`df.collect()`	Retourne les enregistrements sous forme de liste
`df.count()`	Retourne le nombre de lignes dans le DataFrame
`df.describe(*cols)`	Calcule les statistiques descriptives des colonnes numériques
`linesDF.cache()`	Enregistre le DataFrame dans le cache donc pas besoin de réexécuter toutes les transformations et actions. A utiliser si on réutilise souvent le DataFrame.
`aDF.unionAll(bDF)`	Concaténation de deux DataFrame

/!\ ATTENTION /!\

- S'assurer qu'on a assez d'espace dans le programme "driver" pour utiliser `collect()`.
- Ne jamais utiliser `collect()` en production . Préférer `take(n)`.

```
df.groupBy(df.name).agg({'*': 'count'}).collect()
df.groupBy(df.name).count()
df.groupBy().avg().collect()
df.groupBy('name').avg('age', 'grade').collect()
```