Overview of the course

1. Introduction
2. Apache Spark’s basic concepts
3. **Spark SQL**
4. Mllib

Data Frame

- A Data Frame is a distributed collection of data, organised as columns with an associated name.
  - The concept is similar to SQL tables

- The entry point to Spark SQL is the SQLContext class:

Python:

```
from pyspark.sql import SQLContext
sql = SQLContext(sc)
```

Creating a Data Frame

- Data Frames can be loaded from different sources: JSON, JDBC/ODBC and Apache Hive.
  - Check official documentation
- In addition, Spark allows creating data frames from RDDs

```
In [13]: rdd = sc.parallelize([("Maria",35),
("Jose",42), ("Antonia", 25)])
In [14]: people = sql.createDataFrame(rdd)
In [15]: people.show()
+-------+---+
|     _1| _2|
+-------+---+
|  Maria| 35|
|   Jose| 42|
|Antonia| 25|
+-------+---+
```

Providing extra information to Data Frames

```
from pyspark.sql import Row
from pyspark.sql.types import *
rdd = sc.parallelize([("Maria",35), ("Jose",42),
                                        ("Antonia", 25)])
schema = StructType([
                  StructField("name", StringType()),
                  StructField("age",  IntegerType())])
sql = SQLContext(sc)
people = sql.createDataFrame(rdd,schema)
people.show()
+-------+---+
|   name|age|
+-------+---+
|  Maria| 35|
|   Jose| 42|
|Antonia| 25|
+-------+---+
```

--- Common operations and transformations for DF’s

- **agg(*expressions)** returns a new DF with a single row, containing the results of the expressions passed by parameter

```
people.agg(avg(people.col("age")),max(people.col("age")),
min(people.col("age"))).show()

|avg(age)|max(age)|min(age)|
+--------+--------+--------+
|    34.0|      42|      25|
```

- **corr(col1, col2), cov(col1, col2)** returns the correlation/covariance between two columns
- **drop(col)** returns a new DF with the specified column dropped from the original
- **withColumn(name, expr)** returns a new column, given a name and the expression that provides its value:

```
people.withColumn("female",
people.col("name").endsWith("a")).show()

|   name|age|female|
+-------+---+------+
|  Maria| 35|  true|
|   Jose| 42| false|
|Antonia| 25|  true|
```

- Other methods common to RDDs: count, distinct, filter, ...

SQL-like operations on DFs

- **select(*cols)** returns a new DF with only the specified columns:

```
people.select("name").show()
+-------+
|   name|
+-------+
|  Maria|
|   Jose|
|Antonia|
+-------+
```

- **filter(condition), where(condition)** filters the rows given a condition

```
people.where(people.col("age") < 30).show()
+-------+---+
|   name|age|
+-------+---+
|Antonia| 25|
+-------+---+
```

- **groupBy(*columns)** similar to SQL GROUP BY, providing aggregation functions:

```
people.groupBy(people.col("age") % 10).avg().show()
+----------+--------+
|(age % 10)|avg(age)|
+----------+--------+
|         2|    42.0|
|         5|    30.0|
+----------+--------+
```

SQL text queries

```
from pyspark.sql.types import *
sql = SQLContext(sc)
schema = StructType([
            StructField("name",StringType()),
            StructField("age",IntegerType()),
            StructField("passport",StringType())])


students = sql.createDataFrame(
       sc.parallelize([
          ("Jaime",32,"12345-f"),
          ("Maria",19,"22222-g"),
          ("Alex",23,"65432-z")]),chema)

students.registerTempTable("students")
```

SQL text queries (II)

```
schema2 = StructType([
   StructField("passport",StringType()),
   StructField("year",IntegerType())])


enrollments = sql.createDataFrame(
   sc.parallelize([
       ("12345-f",1990),
       ("22222-g",2014),
       ("65432-z",2009)]),schema2)

enrollments.registerTempTable("enrollments")
```

SQL text queries (III)

```
sql.sql("""
     SELECT students.name, enrollments.year AS
enrollment_year
     FROM students, enrollments
     WHERE students.passport = enrollments.passport
     ORDER BY students.name ASC
     """).show()
```

```
+-----+---------------+
| name|enrollment_year|
+-----+---------------+
| Alex|           2009|
|Jaime|           1990|
|Maria|           2014|
+-----+---------------+
```

Hands-on: Google Cluster data analyser

https://github.com/mariomac/patc-spark/tree/master/exercises/2-sql