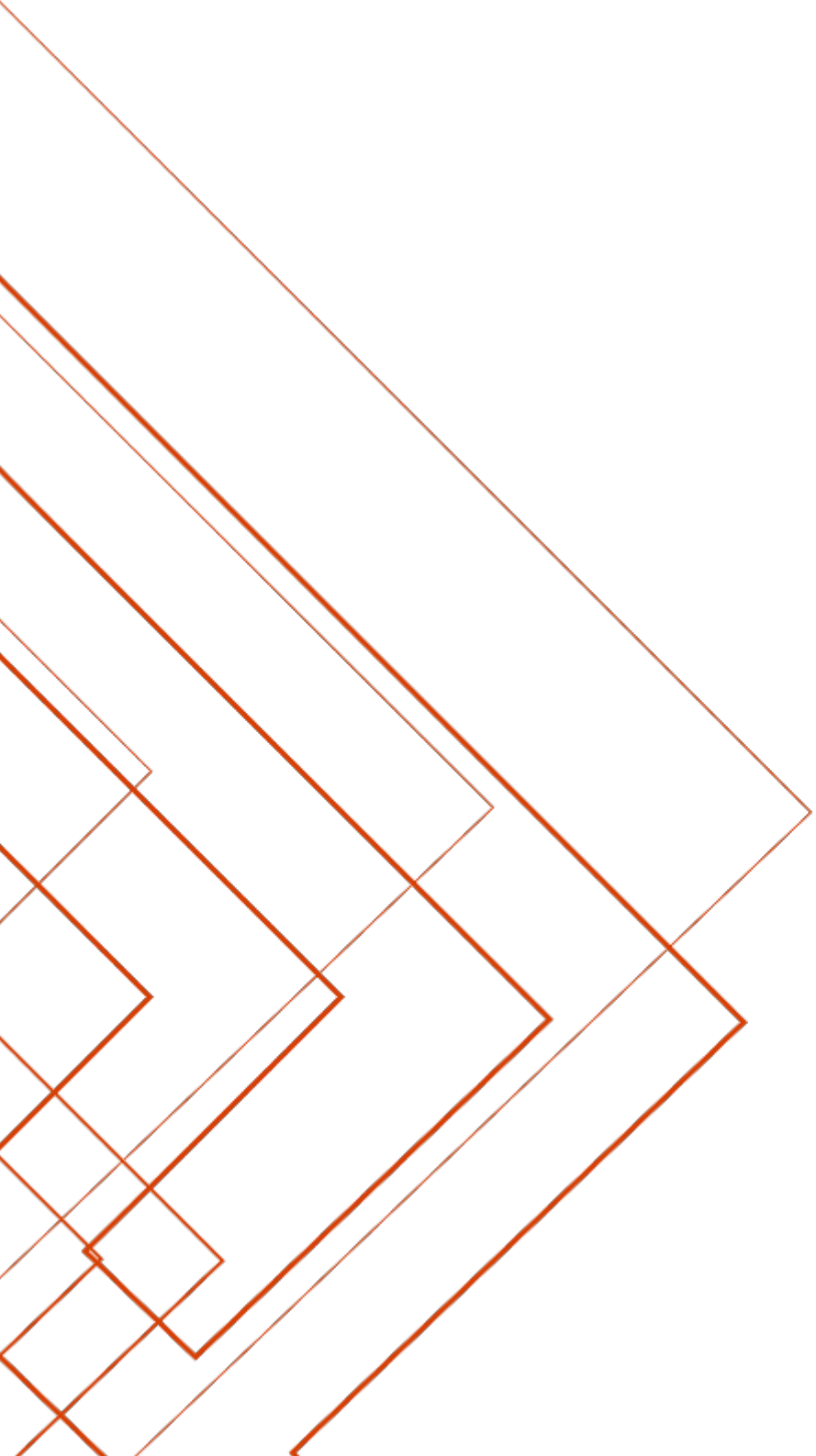


Afi Escuela

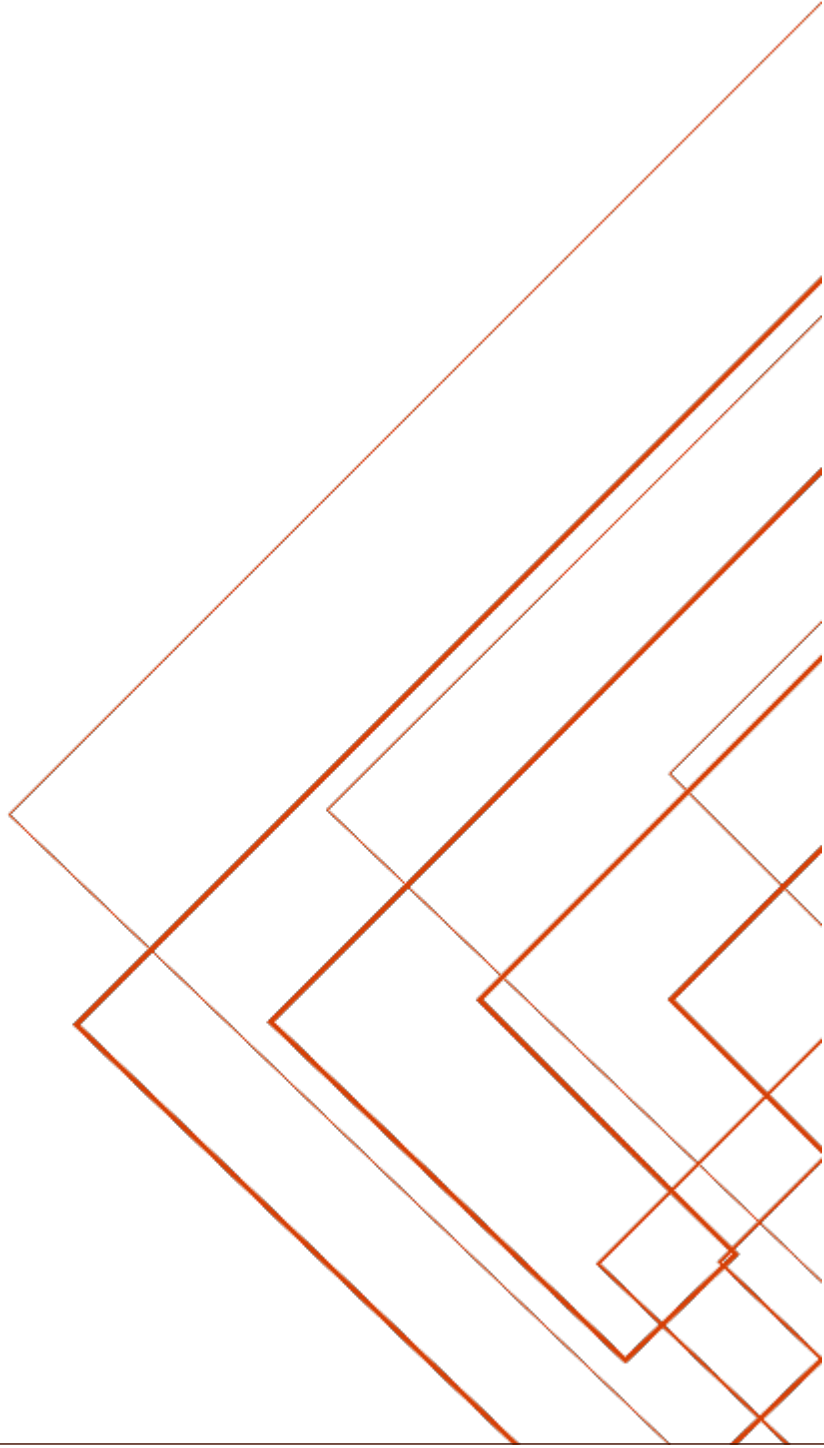
Apache Spark - SQL

Máster en Data Sience y Big Data

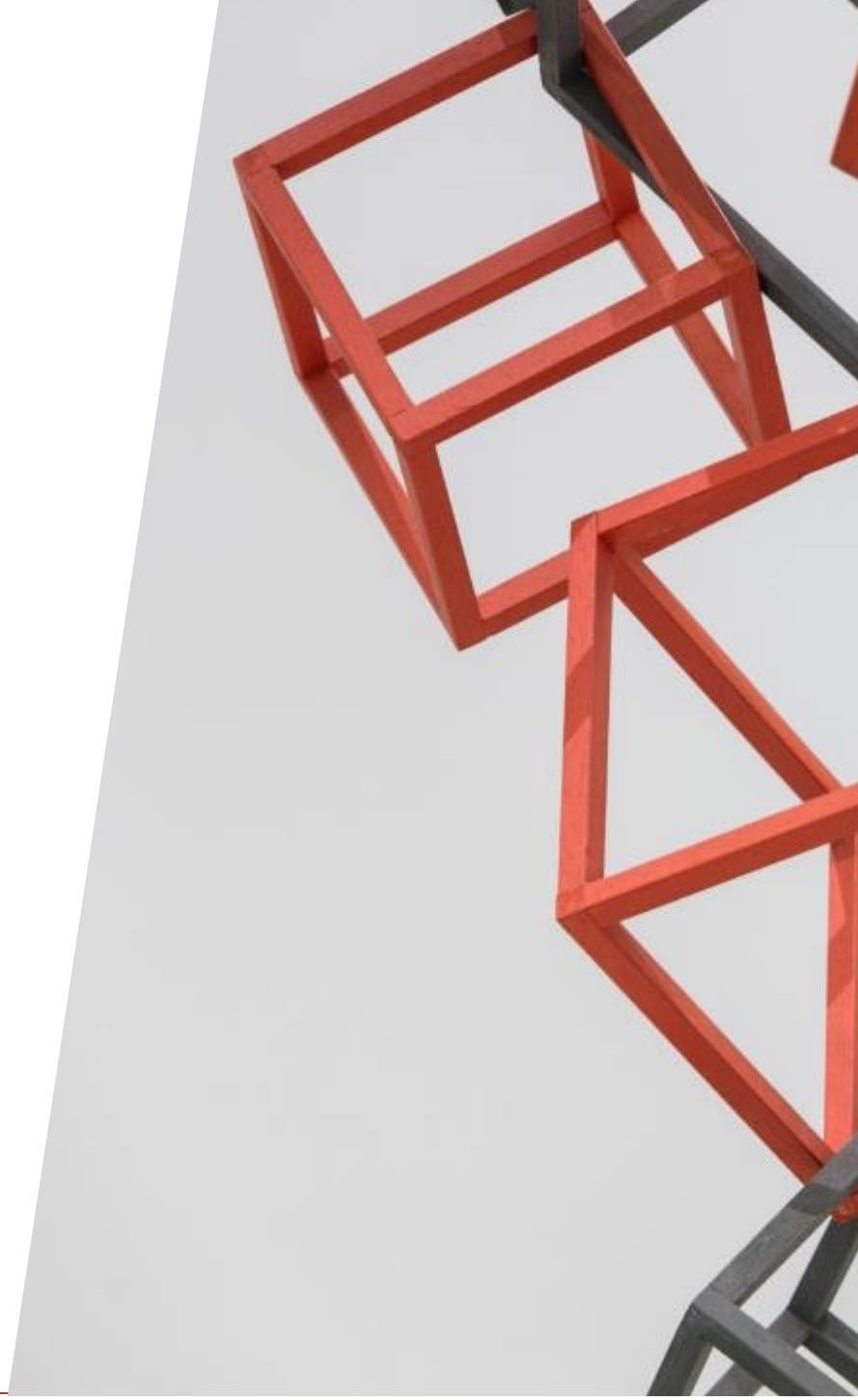
Miguel Ángel Corella
mcorella@geoblink.com

Marzo 2022

Contenido



Introducción



Spark Core, ¿es suficiente?

- Apache Spark ha dado una solución global a los principales problemas que Hadoop presentaba:
  - Velocidad, facilidad de uso, flexibilidad, homogeneidad.

- Su utilidad es clara cuando hablamos de tareas más cercanas al *data engineering*:
  - Carga de (grandes) datos, *parsing*, limpieza/filtrado de información, etc.

- Sin embargo, cuando tratamos de realizar tareas más cercanas al *data science* o *data analysis*, empezamos a detectar ciertas carencias.
  - Aunque más sencillo que con map/reduce, hacer consultas **no es fácil**.
  - Los datos "de base" son desestructurados, debemos **definir la estructura a mano.**
  - Además esta estructura no es persistente, debemos **definirla en cada paso.**
  - Aunque no necesitamos expertos en Java como ocurría en map/reduce, seguimos necesitando **expertos en Python** (hay más analistas que programadores).

¿Qué es Apache Spark SQL?

- Es un módulo de Apache Spark creado para habilitar el **procesamiento de datos estructurados**.

- Nos permitirá realizar todas las **operaciones soportadas por otras librerías** de trabajo sobre datos estructurados (p.e. pandas) abstrayéndonos de su volumen y distribución.

- Incluso, nos permitirá llevar a cabo la **ejecución de consultas SQL** sobre datos estructurados abstrayéndonos de su volumen y distribución.

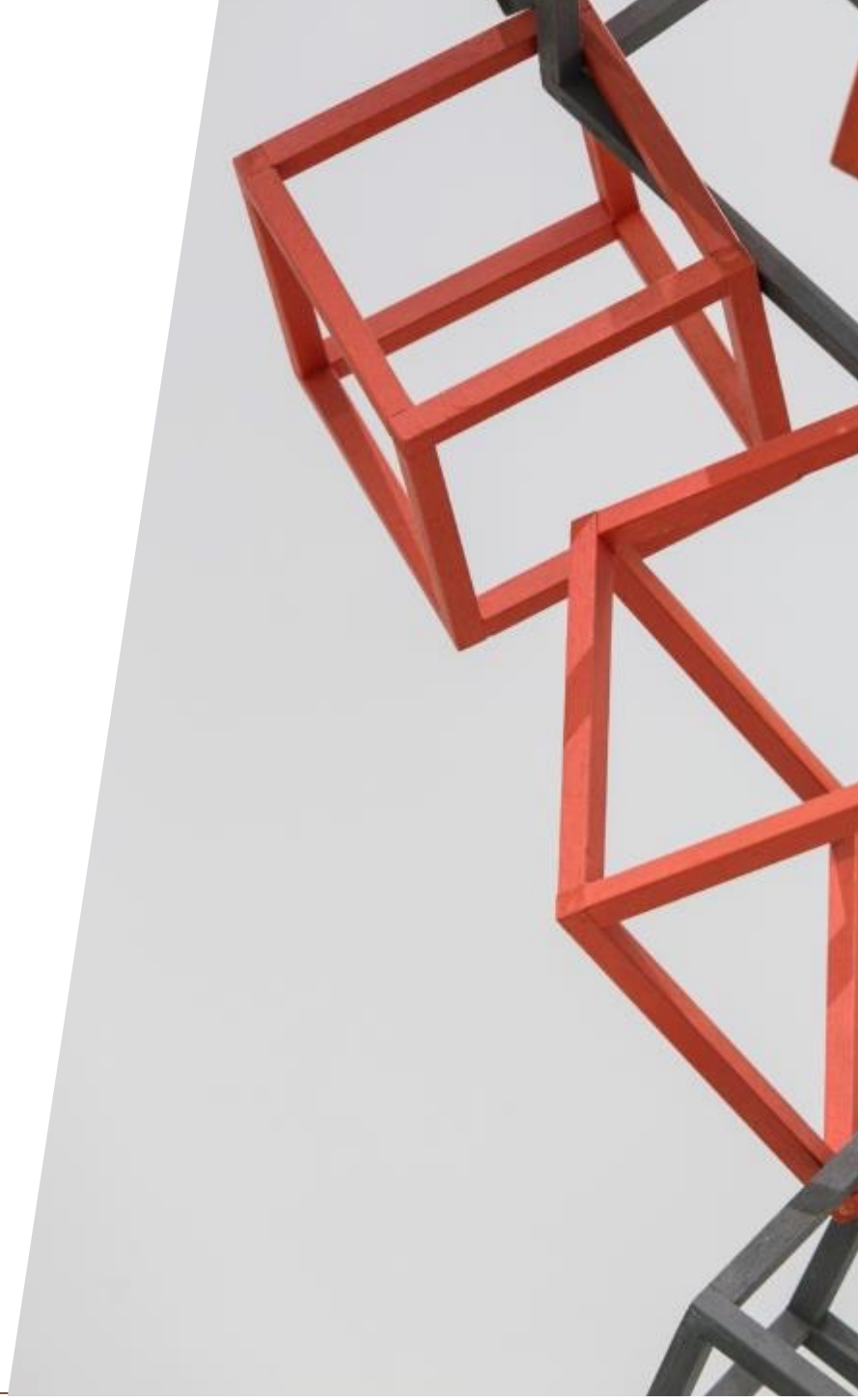
- Realmente, se trata de **una capa de abstracción** ya que, internamente, la ejecución final de tareas la realiza Spark Core.

¿Qué es Apache Spark SQL?

- En Spark SQL, al contrario que con RDD y Pair RDD, los **datos** son almacenados **junto con** información sobre su **estructura**. Esta nueva información habilita:
  - Optimización de scripts para los desarrolladores/analistas: no es necesario "parsear" la información cada vez.
  - Nuevas funciones: que evitan la necesidad de definir funciones para TODO (filtros, ordenaciones, agrupaciones, etc).
  - Optimización de velocidad de ejecución: al conocer a priori la estructura (y tipo) de los datos, la ejecución de tareas se puede optimizar aún más.

- Además, al ser un módulo más de Spark, permite su integración directa con el resto de módulos de forma que podamos obtener "lo mejor" de cada uno.
  - Datos no estructurados y operaciones de bajo nivel: Spark Core.
  - Datos estructurados, consulta y análisis exploratorio: Spark SQL.
  - Análisis estadístico y Machine learning: Spark MLlib.
  - …

SparkSession



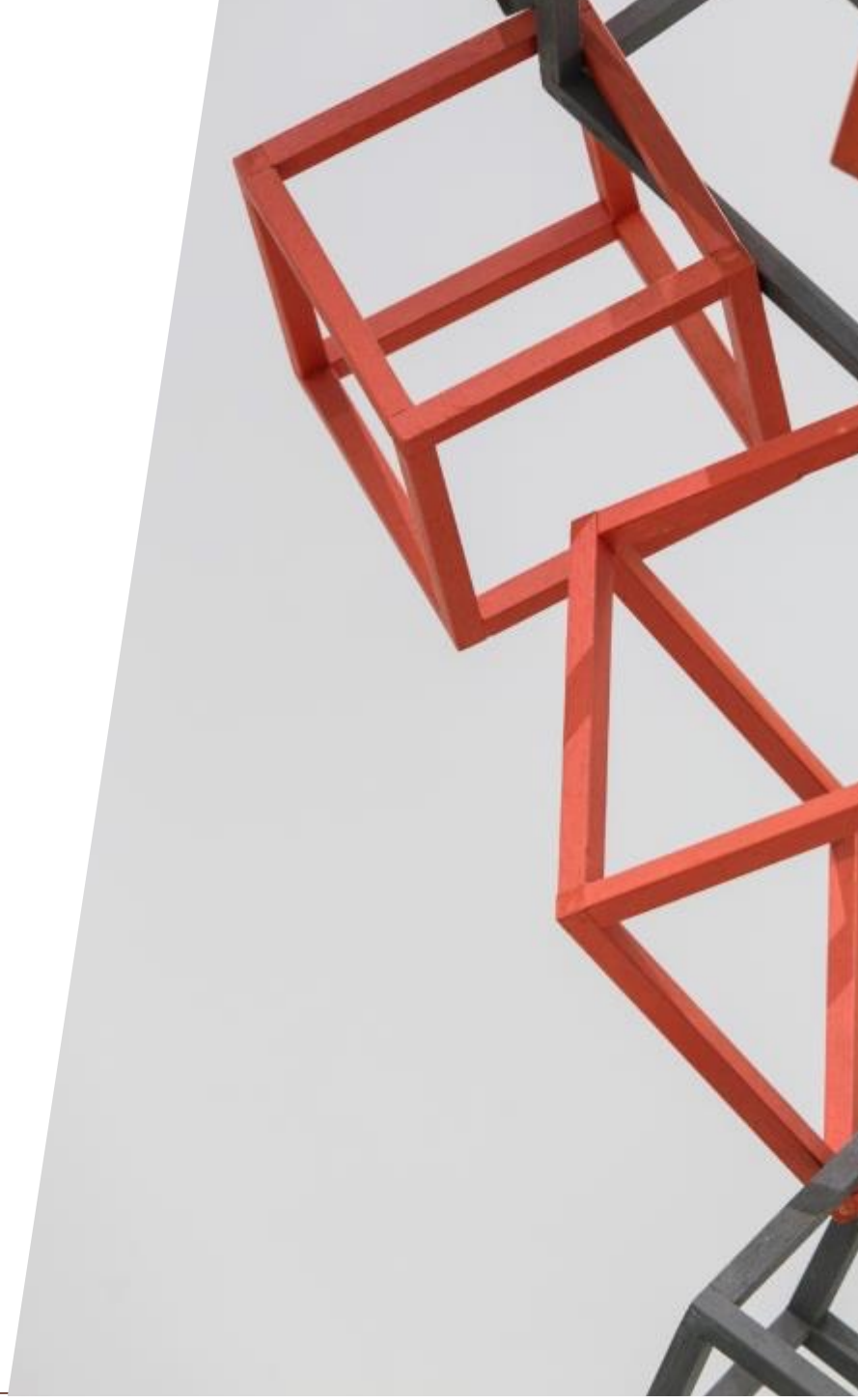
SparkContext vs. SparkSession

- Para poder hacer uso de **Spark Core**, lo primero que hay que hacer es crear un objeto de tipo **SparkContext**.
- Para poder hacer uso de **Spark SQL**, lo primero que hay que hacer es crear un objeto de tipo **SparkSession**.
- Spark SQL está construido "sobre" Spark Core, por lo que un objeto de tipo **SparkSession SIEMPRE contiene** un objeto de tipo **SparkContext**.
- En versiones anteriores de Spark había que crear explícitamente un **SparkContext** y, luego, pasarlo como parámetro en la creación de un **SQLContext o HiveContext**.
- A partir de la versión 2.0, **SparkSession** nos da un único punto de acceso a todo.

Juguemos un poco...

SparkSession.ipynb

DataFrames



DataFrames

- Un DataFrame (o Schema RDD) es, ante todo, un RDD, por lo que comparte todas sus propiedades:
  - **Resilient:** tolerancia a fallos, replicación y particionado.
  - **Distributed:** distribución en diferentes máquinas físicas / *cluster*.
  - **Dataset:** puede contener tanto cualquier tipo de datos básico o compuestos.
  - **In-memory:** se almacena en memoria tanto como sea posible.
  - **Read-only:** el contenido de un RDD no puede modificarse.

- Pero al contrario que los RDD, los datos están **estructurados** (tienen esquema) y **organizados en columnas con nombre** (y tipo, aunque en Python los tipos…).

- Conceptualmente, se puede pensar en DataFrames como:
  - Tablas de base de datos relacionales.
  - DataFrames de R o pandas.

DataFrame Schema

- El *schema* (o esquema) de un DataFrame es una estructura de datos que contiene información sobre la estructura de los datos contenidos en el DataFrame.

- Es decir, contiene los **metadatos** del DataFrame.

- Estos metadatos definen:
  - Los **nombres de las columnas** del DataFrame.
  - Los **tipos de las columnas** del DataFrame.
  - Un **indicador** de si la columna puede tomar o no **valores nulos**.

- A la hora de crear un DataFrame tendremos que definir sus datos y su esquema,
  - Aunque la definición explícita del esquema es opcional (relativamente).

Creación de DataFrames

- Desde un RDD pasando explícitamente un schema:
  - Un schema se define con un objeto de tipo StructType.
  - Un objeto de tipo StructType contiene una colección de objetos de tipo StructFields.
  - Un objeto de tipo StructField contiene un nombre, un tipo y un indicador de nulos.
  - La definición de tipos válidos en Spark se puede encontrar aquí.

- Desde un RDD de objetos Row:
  - Un objeto Row puede verse como un conjunto de pares clave-valor.
  - Spark inferirá el schema del DataFrame a partir de los tipos y campos de cada Row.

- Desde un fichero de datos estructurados:
  - CSV
  - JSON
  - Parquet.
  - ...

Creación de DataFrames

Creación de DataFrames

Operación	Descripción
spark.createDataFrame(rdd, [schema])	Creación de un DataFrame a partir de un RDD (con esquema opcional)
spark.read.[format](<path>)	Creación de un DataFrame mediante lectura directa de un fichero de datos "estructurado"

Juguemos un poco...

DataFrame – Creation.ipynb

Operaciones sobre DataFrames

- Una vez disponemos de un conjunto de datos "estructurado", Spark SQL pone a nuestra disposición un abanico amplio de operaciones para trabajar sobre el mismo.

- Estas operaciones cubrirán las mismas funcionalidades que se pueden hacer con este tipo de estructuras en SQL (al igual que en R y Python).

- En concreto:
  - Inspección de estructura: entender cómo están organizados los datos.
  - Inspección de contenido: visualizar los valores concretos en los datos.
  - Operaciones de "tabla": selección, filtrado, agrupación, ordenación, unión...
  - Operaciones de "columna": referencia, renombrado, transformación...
  - Conversión: transformación en RDDs, objetos de pandas...
  - Persistencia: escritura a disco.

Inspección de estructura

- Spark SQL pone a nuestra disposición herramientas suficientes para poder tener una visión clara de la estructura que siguen nuestros datos:
  - Esquema.
  - Columnas.
  - Tipos.

Inspección de estructura

Operación	Descripción
df.dtypes	Listado de columnas con nombre y tipo
df.schema	Visualización del esquema como StructType
df.printSchema()	Visualización "formateada" del esquema
df.columns	Listado de columnas del DataFrame
df.count()	Número de filas del DataFrame

Inspección de contenido

- Spark SQL pone a nuestra disposición un abanico amplio de herramienta para **recuperar el contenido (total o parcial) de un DataFrame**.

- Estas inspecciones de contenido son las verdades **ACCIONES de Spark SQL**.

Inspección de contenido

Operación	Descripción
df.show(n)	Muestra n filas en formato tabla
df.head(n)	Muestra n filas como lista de objetos Row
df.tail(n)	Muestra últimas n filas como lista de objetos Row
df.take(n)	Muestra n filas como lista de objetos Row
df.first()	Muestra la primera fila como objeto Row
df.describe()	Cálculo de estadísticos básicos

Operaciones de “columna”

- Spark SQL trabaja con datos estructurados por lo que TODAS las operaciones se basan en el uso de columnas.

Referencia

Operación	Descripción
df.<col>	Referencia (sólo referencia) a columna como atributo
df[‘<col>’]	Referencia (sólo referencia) a columna por nombre

Modificación directa

Operación	Descripción
col.alias(<new>)	Renombrado/alias de columna
col.cast(<type>)	Cambio de tipo de columna
col.substr(<start>, <length>)	Substring de cadenas

Operaciones de “columna”

- Spark SQL nos da acceso a funciones “básicas” de SQL a través de su submódulo “functions”.

> from pyspark.sql import functions as F

Aplicación de funciones

Operación	Descripción
F.when(<cond>, <val>).otherwise(<val>)	Case-When-Else sobre columna
F.abs(<col>)	Valor absoluto de una columna
F.sqrt(<col>)	Raíz cuadrada de una columna
F.pow(<col>, <exp>)	Potenciación de una columna
F.round(<col>, <pos>)	Redondeo de decimales de una columna

Operaciones de “columna”

Condicionales

Operación	Descripción
col.isNull() / col.isNotNull()	Condición sobre campos vacíos / no vacíos
col.isin(<list>)	Condición sobre coincidencia en lista
col.like(<pattern>)	Condición sobre patrón de cadena
col.contains() / col.startswith() /col.endswith()	Condición sobre match parcial de cadena
col.between(<low>, <high>)	Condición sobre rango

Ordenación

Operación	Descripción
col.asc() / col.desc()	Ordenación ascendente / descendente
col.asc_nulls_first() / col.desc_nulls_first()	Ordenación con prioridad de nulos
col.asc_nulls_last() / col.desc_nulls_last()	Ordenación con prioridad de no nulos

Operaciones de “tabla”

- Spark SQL nos permite hacer todas las operaciones sobre DataFrames que nos permiten hacer herramientas de análisis de datos como pandas o RDBMS.

Operaciones de “tabla”

Operación	Descripción
df.select(<selection>)	Selección de un conjunto de columnas.
df.where(<condition>)	Filtrado de los contenidos en base a condiciones sobre columnas.
df.orderBy(<columns>,<dirs>)	Ordenación de contenidos en base a columnas.
df.groupBy(<columns>)	Agrupación de contenidos en base a valores de columnas.
df.join(<other>, <on>, <how>)	Cruce de contenidos en base a columnas compartidas.
df.withColumn(<new>, <col_op>)	Creación de nueva columna desde operación de columnas
df.withColumnRenamed(<old>, <new>)	Renombrado de columna

Conversión

- Spark SQL facilita la conversión de DataFrames en otras estructuras de datos tanto dentro del mismo Spark como para su posterior trabajo en "local".

De Spark SQL a Spark Core

Operación	Descripción
df.rdd	Convierte un DataFrame de Spark en un RDD de Rows

De Spark SQL a pandas

Operación	Descripción
df.toPandas()	Convierte un DataFrame de Spark en uno de pandas

Persistencia

- Del mismo modo que podemos iniciar el trabajo con un DataFrame a partir de un fichero, Spark SQL nos permite volcar el contenido de un DataFrame a ficheros persistentes.

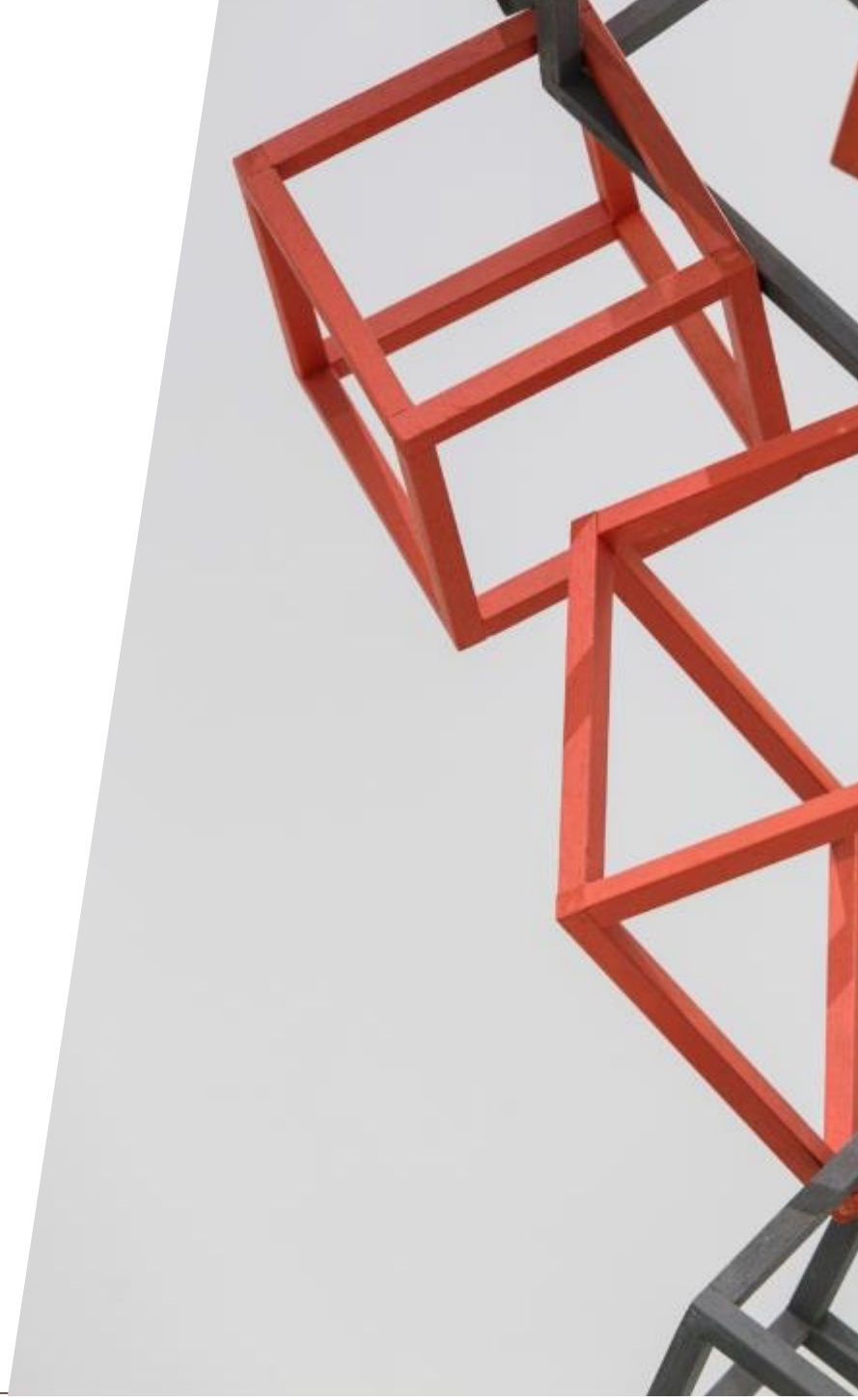
Persistencia a disco

Operación	Descripción
df.write.[format](<path>)	Salva un DataFrame a fichero

Juguemos un poco...

DataFrame – Operations.ipynb

Consultas SQL



Consulta de datos en Spark SQL

- Spark SQL pone a nuestra disposición **dos formas muy distintas de consultar el contenido** de DataFrames:
  - Mediante la utilización de funciones especificas de consulta.
  - Mediante la ejecución de consultas SQL.

- En el caso de las funciones específicas de consulta, Spark SQL nos ofrece **una función para cada posible fragmento (*clause*) de una query SQL**. Así, tendremos funciones específicas para selección de columnas, filtrado, ordenación, agregación…

- En el caso de SQL, únicamente tendremos que **"registrar" nuestro(s) DataFrame(s)** como tabla/vista lógica de datos y podremos **utilizar sintáxis SQL (casi completa)** para definir nuestras consultas.

- Inicialmente, Spark SQL sólo ofrecía funciones específicas, pero, poco a poco, ha ido dando cada vez más soporte a la utilización directa de SQL (por motivos obvios).

Consultas SQL

Registro de "tablas"

Operación	Descripción
df.createTempView(<name>)	Registra el DataFrame con el nombre <name> para su posterior uso como tabla SQL
df.createOrReplaceTempView(<name>)	Registra o sustituye el DataFrame con el nombre <name> para su posterior uso como tabla SQL

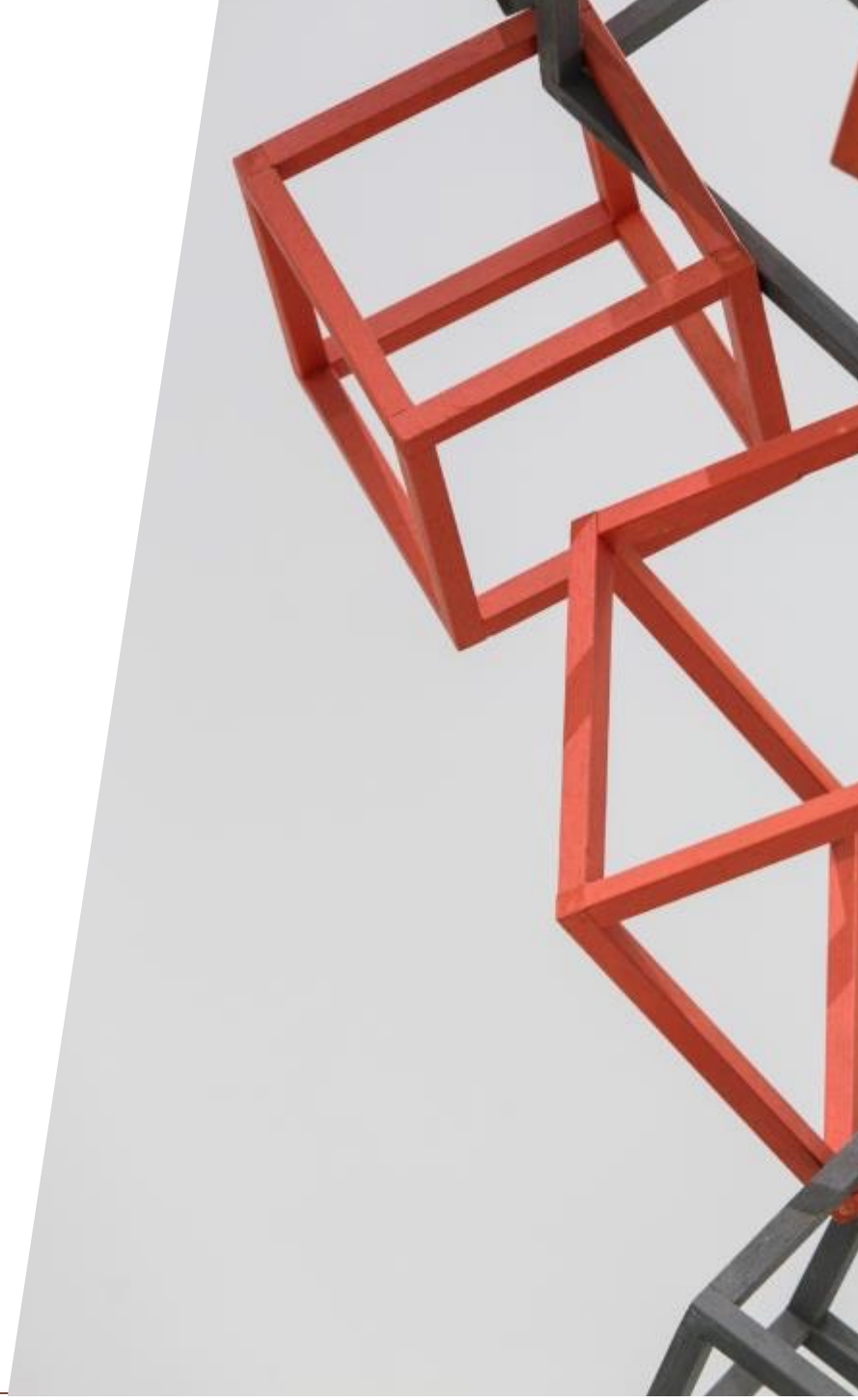
Ejecución de consultas

Operación	Descripción
spark.sql(<query>)	Ejecuta la <query> sobre las tablas registradas en la sesión

Juguemos un poco...

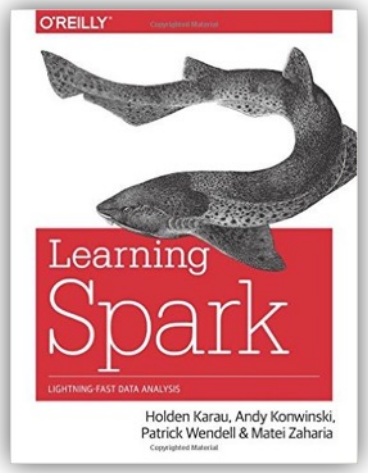
DataFrame – SQL Queries.ipynb

Referencias



Referencias

- Documentación oficial:
  - https://spark.apache.org
- Tutoriales online:
  - https://www.tutorialspoint.com/apache_spark/
- Libros:



Afi Escuela