INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ACADEMIA INGENIERÍA DE SOFTWARE

Unidad de Aprendizaje: Big Data

Profesor: Ing. Tania Rodriguez Sarabia

Práctica no. 1: ETL con pyspark

ALUMNO:

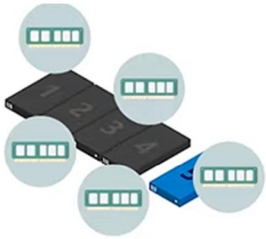
GRUPO:

FECHA DE ENTREGA:

Cuando queremos hacer un análisis de datos sea hallar algún estadístico, hacer una visualización, un filtrado o entrenar a un modelo de machine learning, usualmente cargaremos nuestros datos desde un archivo almacenado en nuestra computadora hacia un DataFrame de pandas, aquí gracias a la magia de pandas vamos a poder hacer el análisis que queramos, esta rapidez de pandas se debe a que los dataframes se cargan en la memoria RAM de sus computadoras una sola vez, luego de esta carga inicial, cualquier análisis u operación ejecutada sobre el DataFrame se hace con los datos ya cargados en la memoria RAM, lo interesante es que las memorias RAM son muy rápidas al momento de operar sobre datos que ahí se encuentran. Pero ¿qué pasa cuando mis datos no entran en la memoria RAM? Esto podría suceder muy fácilmente ya que las computadoras de hoy pueden tener entre 8 y 16 GB de memoria RAM, en este punto podríamos obtener un error si queremos cargar todo en un dataframe de pandas

Ahora nos tocará implementar una solución en donde nuestro código donde nuestro código lea y opere sobre nuestros datos por partes, estas múltiples lecturas al disco duro matarán la eficiencia y rapidez en nuestro proceso pues a comparación de la memoria RAM leer datos del disco duro es de 10 a 100 veces mas lento dependiendo de la calidad del disco.

Sin embargo, si tenemos mas de una maquina a nuestra disposición, por cada máquina tendríamos una memoria RAM adicional que podríamos utilizar para que este procesamiento siga siendo rápido y por sobre todo que soporte grandes cantidades de datos, con esta premisa nacen algunos problemas ¿cómo dividimos los datos en las memorias RAM de varias máquinas? ¿Cómo controlamos que dato están en cual máquina? ¿Cómo unimos los datos cuando queremos hacer algún análisis como una agrupación?

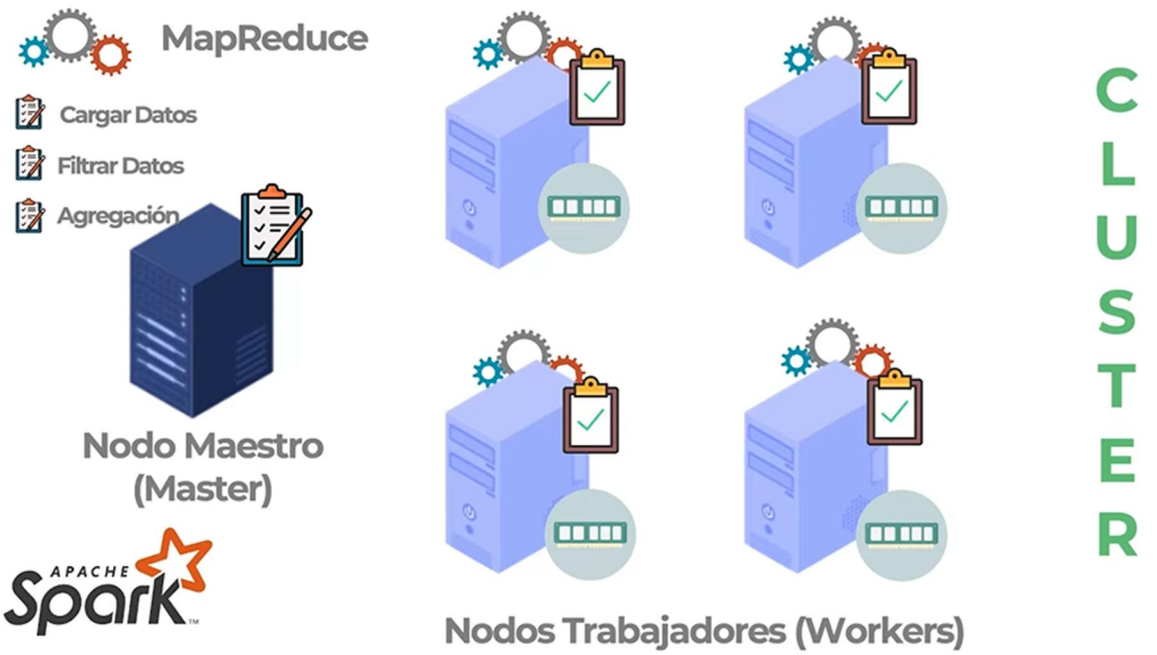


¿Dividir los Datos?

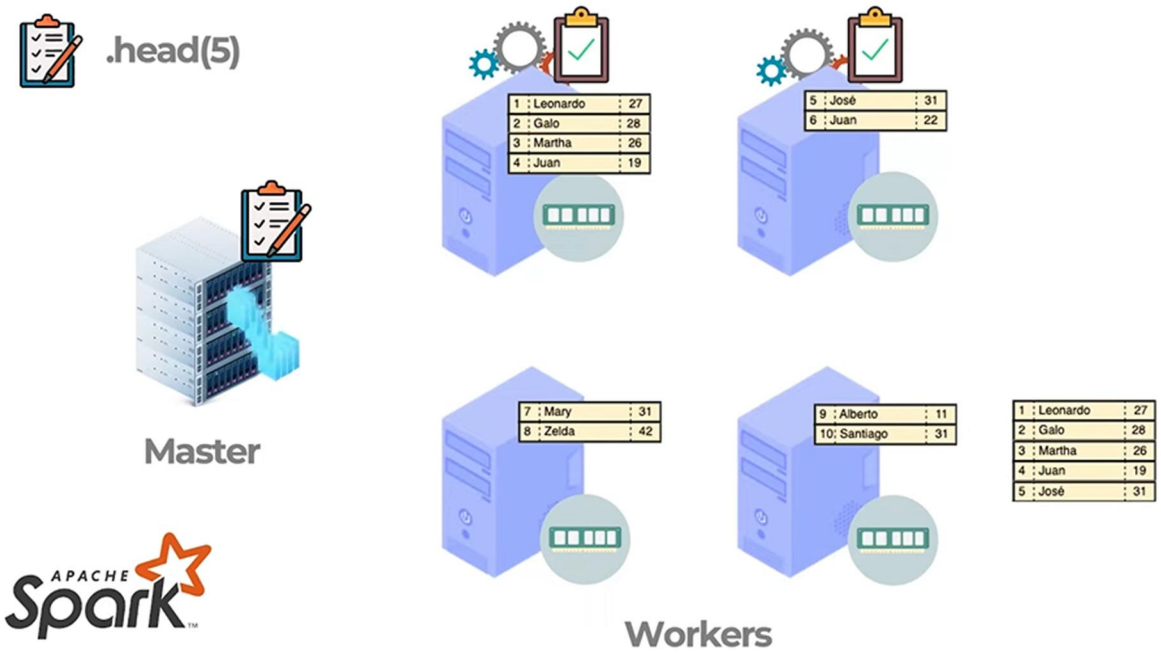
¿Control de los Datos?

¿Unión de los datos?

Con estos problemas nace Apache Spark, que es un framework para procesar grandes cantidades de datos cuya idea principal es distribuir los datos en las memorias RAM de varias máquinas y que cada máquina se encargue del procesamiento de la parte de los datos que esta en ella. A este conjunto de computadoras se les denomina clusters y a cada computadora en el cluster se le llama nodo, por cada cluster existe un nodo maestro o master node y varios nodos trabajadores o workers. El master node se encarga de recibir las tareas que tienen que ejecutarse y dividir esas tareas entre los nodos trabajadores los cuales van a ejecutar estas tareas en paralelo sobre su parte de los datos utilizando el paradigma de programación MapReduce, que es un divide y conquista para grandes cantidades de datos. Las tareas pueden ser Cargar los datos, Filtrar los datos, hacer alguna Agregación o incluso entrenar un modelo de machine learning.

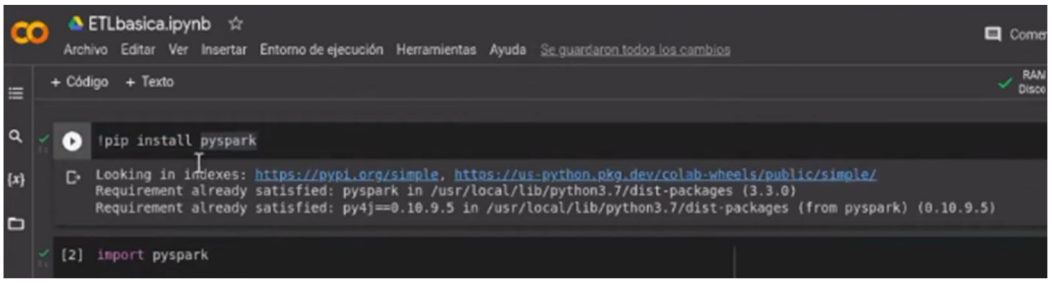


Cuando los datos se cargan estos se distribuyen idealmente de manera uniforme a lo largo de las memorias RAM de los nodos workers y el nodo maestro conoce hasta cierto punto cómo se encuentran hechas estas divisiones de los datos de modo que el nodo maestro puede saber a que nodos trabajadores asignar ciertas tareas. Por ejemplo, si el usuario nos pide mostrar los 5 primeros registros de un dataset en este ejemplo solo el nodo trabajador 1 y 2 van a recibir tareas, evitando el trabajo innecesario de los otros nodos, los resultados de cada tarea son retornados por cada worker directamente a la aplicación, donde existe un ente que se encarga de unirlos y presentarlos al usuario.

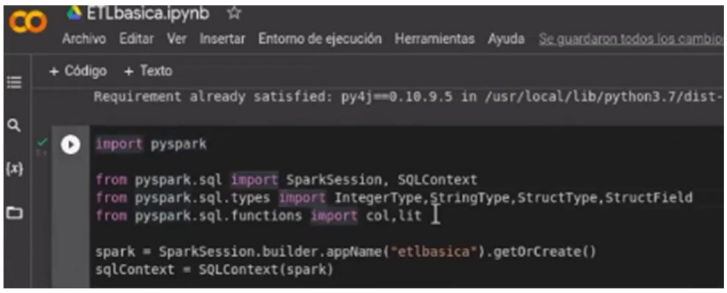


Cabe aclarar que Spark es un framework, esto quiere decir que lo podemos utilizar en muchos lenguajes de programación incluyendo Python con la librería de pyspark, siempre y cuando tengamos un cluster de computadoras con spark instalado y configurado, lo cual puede llegr a ser muy complejo. Pero por suerte, existen algunos proveedores de servicios en la nube como AWS que nos dan herramientas que con un par de clicks, ya tenemos configurado un cluster de máquinas para ejecutar nuestro código spark sobre nuestros datos.

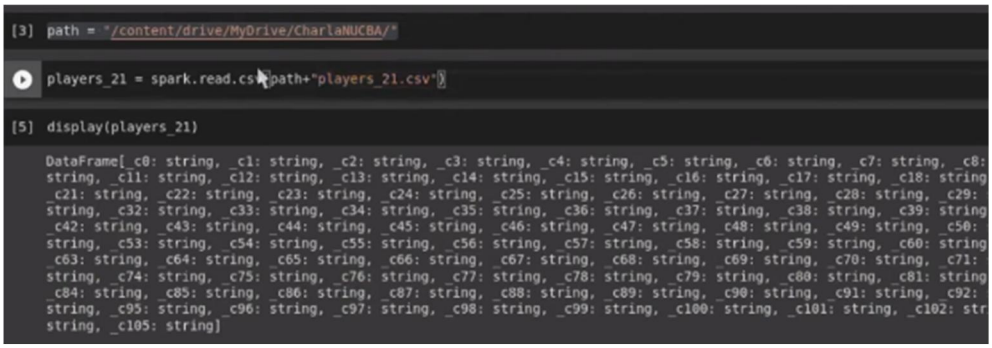
Vamos a haga un proceso de ETL en Spark, primero que nada y por simplicidad vamos a abrir el Google colab, que es un entrono que permite ejecutar código Python en un navegador sin necesidad de configuraciones avanzadas. Primero instalamos todo lo necesrio para trabajar con pyspark.



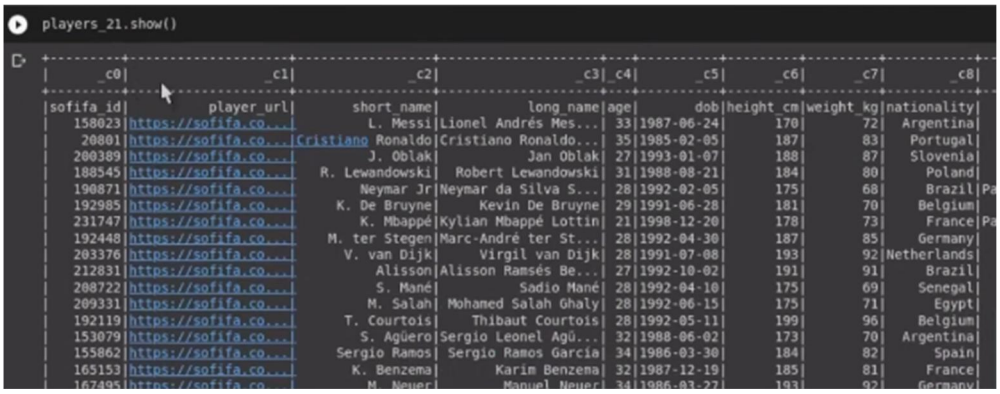
Una vez que tengamos todo el ambiente preparado, importamos todas las herramientas que vamos a usar. También vamos a crear una sesión de spark y también vamos a inicializar nuestro contexto de sql.



Ahora le vamos a indicar la ruta del archivo que se vamos a trabajar, y también vamos a crear una variable con los datos de un csv llamado players_21.csv, en este caso es csv, pero pyspark puede igualmente consumir otros tipos de datos como parquet, json, etc. Luego, vamos a mostrar el csv con display(), esto mostrará el DataFrame tal cual está constituido



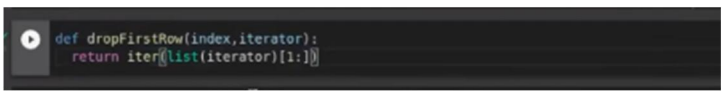
También hay otras formar de mostrar el archivo csv como por ejemplo con el comando show(), que lo que hará es mostrar de manera ordenada las columnas.



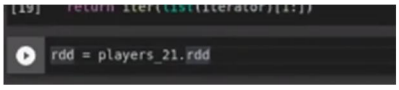
Ya hemos hecho la carga de los datos, pero nos damos cuenta de que hay un montón de datos que no nos sirven, como URL, además los encabezados de las columnas (_c0, _c2, etc.) tampoco nos sirven, también el id por ejemplo no lo vamos a necesitar.

Primero vamos a sobrescribir mi variable players_21, indicando solo las columnas que deseamos mantener con select.

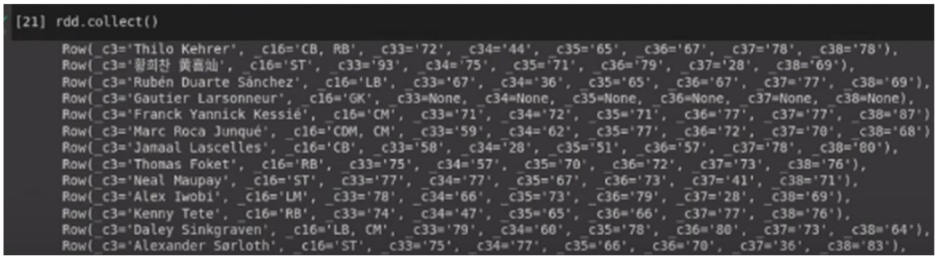
Luego definimos una función para quitar el encabezado (_c0, _c1, _c2, etc.) y queremos el nombre que yo quiero, osea el de la segunda fila (soffifa_id, player_url, short_name, long_name, etc.)



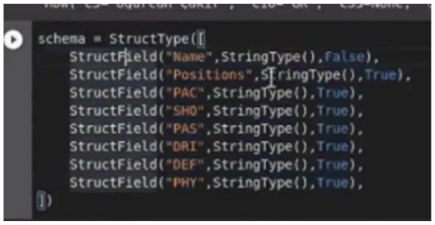
Luego creamos un RDD a partir del dataframe.



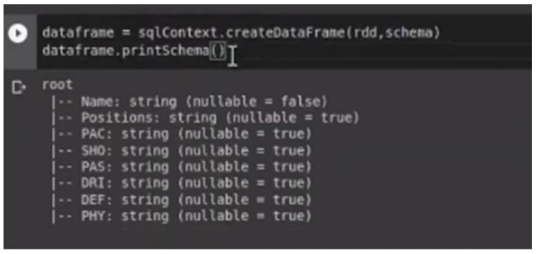
Luego hacemos un collect para ver qué información trae el rdd.



Ahora vamos a convertir nuestro rdd, nuevamente a un dataframe, así que primero creamos el esquema. Para ello definimos el StructField de cada una de las columnas, su tipo y si aceptan valores nulos o no.



Ahora le decimos al rdd que va a hacer un dataframe utilizando el esquema que hicimos anteriormente con el esquema que hicimos anteriormente.



Luego volvemos a mostrar nuestro dataframe, como va quedando hasta ahora

```
dataframe.show()
```

```
+--------------------+----------+----+----+----+----+----+----+
|                Name| Positions| PAC| SHO| PAS| DRI| DEF| PHY|
+--------------------+----------+----+----+----+----+----+----+
|Lionel Andrés Me...|RW, ST, CF|  85|  92|  91|  95|  38|  65|
|Cristiano Ronaldo...|    ST, LW|  89|  93|  81|  89|  35|  77|
|           Jan Oblak|        GK|null|null|null|null|null|null|
|  Robert Lewandowski|        ST|  78|  91|  78|  85|  43|  82|
|Neymar da Silva S...|   LW, CAM|  91|  85|  86|  94|  36|  59|
|     Kevin De Bruyne|   CAM, CM|  76|  86|  93|  88|  64|  78|
|Kylian Mbappé Lottin|ST, LW, RW|  96|  86|  78|  91|  39|  76|
|Marc-André ter St...|        GK|null|null|null|null|null|null|
|      Virgil van Dijk|        CB|  76|  60|  71|  71|  91|  86|
|Alisson Ramsés Be...|        GK|null|null|null|null|null|null|
|          Sadio Mané|        LW|  94|  85|  80|  90|  44|  76|
| Mohamed Salah Ghaly|        RW|  93|  86|  81|  90|  45|  75|
|    Thibaut Courtois|        GK|null|null|null|null|null|null|
|Sergio Leonel Agü...|        ST|  78|  90|  77|  88|  33|  73|
| Sergio Ramos García|        CB|  71|  70|  76|  73|  88|  85|
|       Karim Benzema|    CF, ST|  74|  85|  81|  86|  40|  76|
|        Manuel Neuer|        GK|null|null|null|null|null|null|
|Carlos Henrique V...|       CDM|  65|  73|  76|  72|  86|  91|
|Ederson Santana d...|        GK|null|null|null|null|null|null|
|Raheem Shaquille ...|    LW, RW|  93|  81|  79|  90|  45|  67|
```

Si notamos en los datos hay muchos valores nulos que deseamos quitar, para ello utilizamos la función dropna

```
dataframe = dataframe.dropna()
```

También vamos a castear o convertir el tipo de columnas a entero ya que están como strings.

```
dataframe = dataframe.withColumn("PAC",col("PAC").cast(IntegerType()))
dataframe = dataframe.withColumn("SHO",col("SHO").cast(IntegerType()))
dataframe = dataframe.withColumn("PAS",col("PAS").cast(IntegerType()))
dataframe = dataframe.withColumn("DRI",col("DRI").cast(IntegerType()))
dataframe = dataframe.withColumn("DEF",col("DEF").cast(IntegerType()))
dataframe = dataframe.withColumn("PHY",col("PHY").cast(IntegerType()))
```

Ahora vamos a imprimir el esquema y vemos que ya están como enteros

```
dataframe.printSchema()
```

```
root
 |-- Name: string (nullable = false)
 |-- Positions: string (nullable = true)
 |-- PAC: integer (nullable = true)
 |-- SHO: integer (nullable = true)
 |-- PAS: integer (nullable = true)
 |-- DRI: integer (nullable = true)
 |-- DEF: integer (nullable = true)
 |-- PHY: integer (nullable = true)
```

Luego vamos a hacer otra columna llamada medio (mean) y hacemos una suma de cada una de estas columnas y lo dividimos entre el número de columnas.

```
[30] dataframe = dataframe.withColumn('Mean', ((col('PAC') + col('SHO') + col('PAS') + col('DRI') + col('DEF') + col('PHY')) / lit(6)))
```

Ahora volvemos a mostrar todo el dataframe.

```
dataframe.show()
```

```
+--------------------+----------+----+----+----+----+----+----+------------------+
|                Name| Positions| PAC| SHO| PAS| DRI| DEF| PHY|              Mean|
+--------------------+----------+----+----+----+----+----+----+------------------+
|Lionel Andrés Mes...|RW, ST, CF|  85|  92|  91|  95|  38|  65|77.66666666666667|
|Cristiano Ronaldo...|    ST, LW|  89|  93|  81|  89|  35|  77|77.33333333333333|
|           Jan Oblak|        GK|null|null|null|null|null|null|              null|
|  Robert Lewandowski|        ST|  78|  91|  78|  85|  43|  82|76.16666666666667|
|Neymar da Silva S...|   LW, CAM|  91|  85|  86|  94|  36|  59|75.16666666666667|
|     Kevin De Bruyne|   CAM, CM|  76|  86|  93|  88|  64|  78|80.83333333333333|
|Kylian Mbappé Lottin|ST, LW, RW|  96|  86|  78|  91|  39|  76|77.66666666666667|
|Marc-André ter St...|        GK|null|null|null|null|null|null|              null|
|      Virgil van Dijk|        CB|  76|  60|  71|  71|  91|  86|75.83333333333333|
|Alisson Ramsés Be...|        GK|null|null|null|null|null|null|              null|
|          Sadio Mané|        LW|  94|  85|  80|  90|  44|  76|78.16666666666667|
| Mohamed Salah Ghaly|        RW|  93|  86|  81|  90|  45|  75|78.33333333333333|
|    Thibaut Courtois|        GK|null|null|null|null|null|null|              null|
|Sergio Leonel Agü...|        ST|  78|  90|  77|  88|  33|  73|73.16666666666667|
| Sergio Ramos García|        CB|  71|  70|  76|  73|  88|  85|77.16666666666667|
```

Creamos un output un archivo csv llamado output1.csv

Hemos hecho la parte de carga y transformación de los datos, el resultado es un archivo csv que puede usarse para cargarse en la nube, para análisis de datos o lo que se requiera, todo esta en csv ya tratado.

Por hacer

Ahora vamos a aprender a cargar, explorar y limpiar los datos usando PySpark, el archivo datos_succios.csv contiene errores como: Valores nulos o faltantes, Columnas con nombres incorrectos, Tipos de datos mezclados, Duplicados, Errores tipográficos. Tu tarea será:

1. Cargar los datos en un DataFrame de Spark
2. Renombrar columnas con errores tipográficos
3. Eliminar duplicados
4. Reemplazar valores incorrectos
5. Convertir tipos de datos
6. Normalizar formatos de fecha
7. Cargar los datos limpios en “datos_limpios.csv”