

# Máster en Big Data

---

## Tecnologías de Almacenamiento

### 9. Hands-On: Spark

PRESENTADO:

JOSE DAVID ANGULO GARCIA

ALBERT RIPOLL

## Índice

1. Introducción.....	3
2. Entorno .....	3
3. Inspección de los datos locales .....	3

# 1. Introducción

El objetivo de este Hands-On es el de familiarizarse con una de las librerías más populares del framework de Spark como es SparkSQL

# 2. Entorno

Para la realización de los ejercicios se va a utilizar *spark-shell* en scala o python ya que nos proporciona un entorno muy dinámico para la introducción de funciones y nos permite recibir una respuesta inmediata.

Para ello, nos descargaremos Spark y levantaremos una *spark-shell* en local.

Sigue las siguientes instrucciones si es necesario:

<https://sparkbyexamples.com/spark/install-apache-spark-on-mac/>

<https://sparkbyexamples.com/spark/apache-spark-installation-on-windows/>

<https://sparkbyexamples.com/spark/spark-installation-on-linux-ubuntu/>

**Descarga e instalación del Spark y levantamiento de spark-shell en local.**

## 1) Descargas.

<https://www.oracle.com/java/technologies/downloads/#jdk17-windows>

Se descarga el “x64 Compressed Archive” seleccionando opciones JDK17 y Windows.


JDK 22	JDK 21	JDK 17	GraalVM for JDK 22	GraalVM for JDK 21	GraalVM for JDK 17
JDK Development Kit 17.0.11 downloads					
JDK 17 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC).					
JDK 17 will receive updates under the NFTC, until September 2024. Subsequent JDK 17 updates will be licensed under the Java SE OTN License (OTN) and production limited free grants of the OTN license will require a fee.					
Linux	macOS	Windows			
			Product/file description	File size	Download
			x64 Compressed Archive	172.83 MB	<a href="https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip">https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip</a> ( sha256)

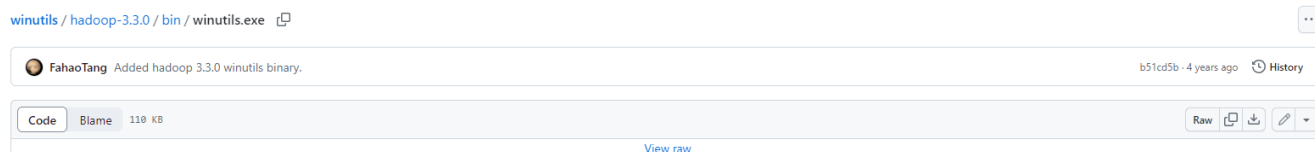
<https://spark.apache.org/downloads.html>

Se descarga spark con enlace de opción 3

## Download Apache Spark™

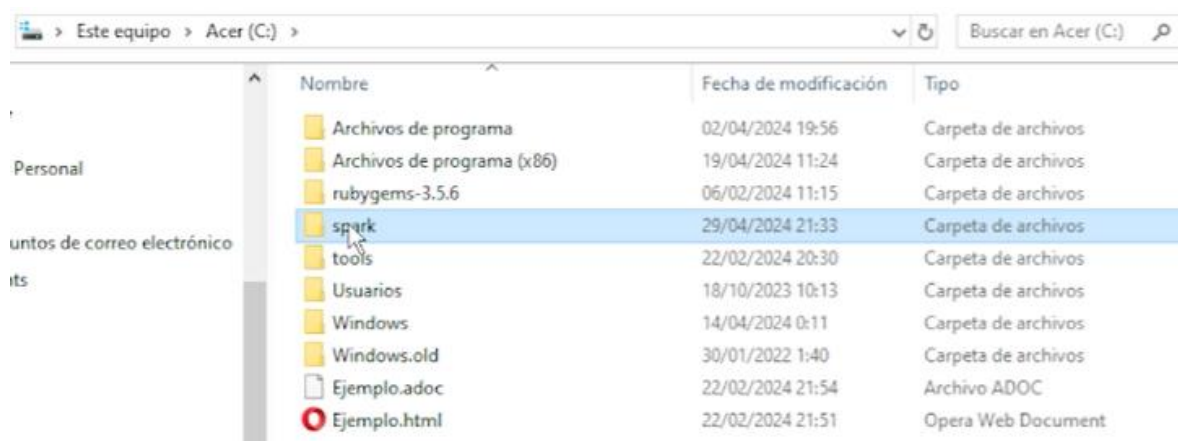
1. Choose a Spark release:
2. Choose a package type:
3. Download Spark: [spark-3.5.1-bin-hadoop3.tgz](#)
4. Verify this release using the 3.5.1 [signatures](#), [checksums](#) and [project release KEYS](#) by following these [procedures](#).

Se descarga winutils con el icono  de la derecha

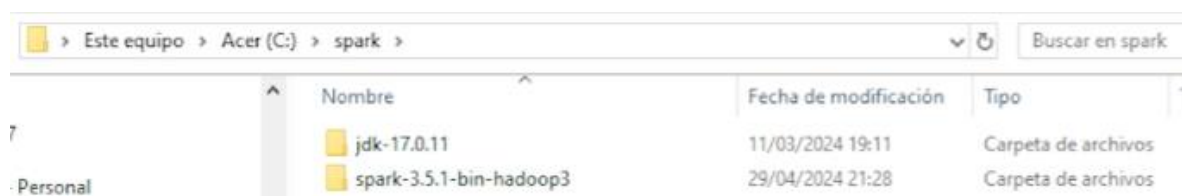


## 2) Colocación de las descargas.

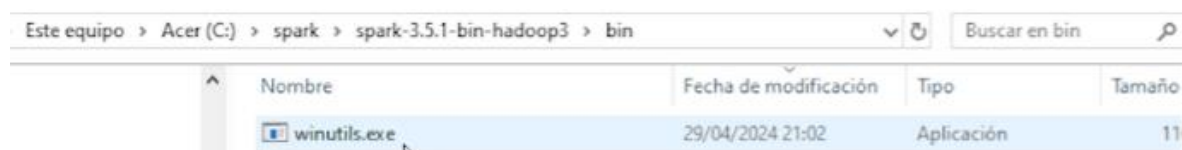
En C: Acer se crea una carpeta llamada “spark”



Dentro de esta carpeta “spark” hemos situado las dos carpetas “jdk-17.0.11” y “spark-3.5.1-bin-hadoop3” de las descargas descomprimidas.



Dentro de “spark-3.5.1-bin-hadoop3” y allí dentro de la carpeta “bin” hemos situado el ejecutable “winutils.exe”.

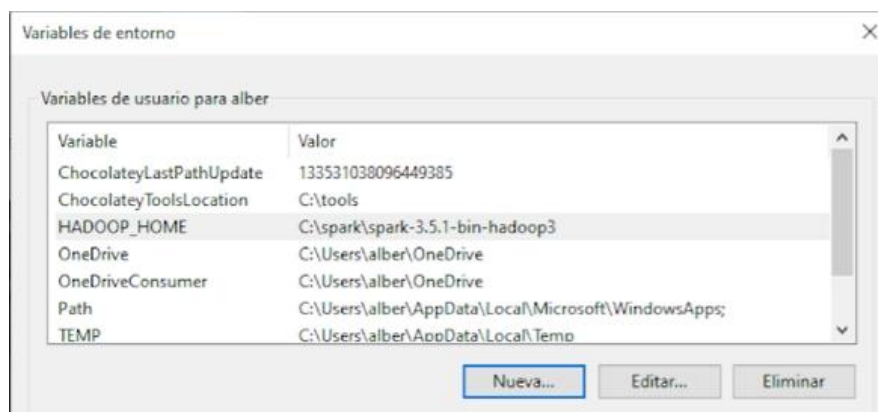


## 3) Editar variables de entorno de esta cuenta

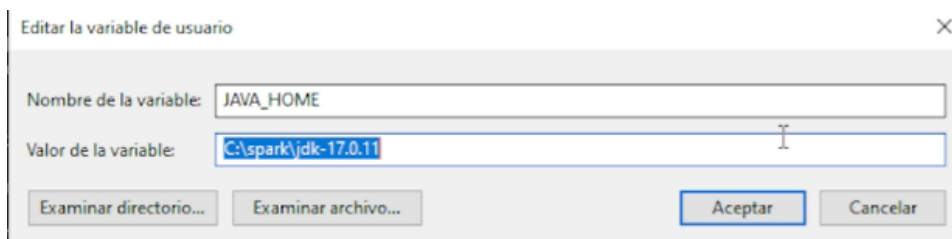
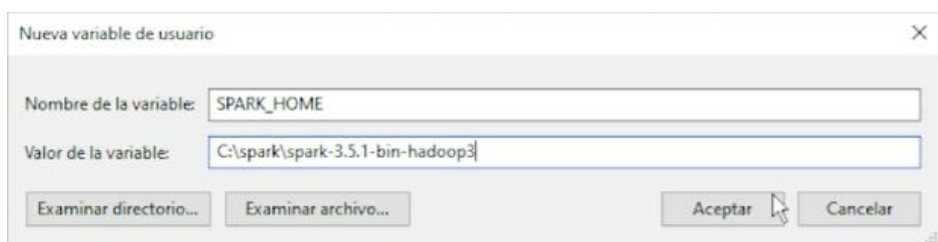
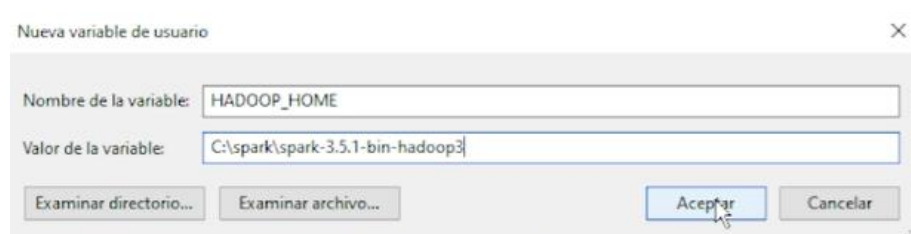
Hemos buscado en Windows la opción “Editar variables de entorno de esta cuenta”



Y allí “Nueva”



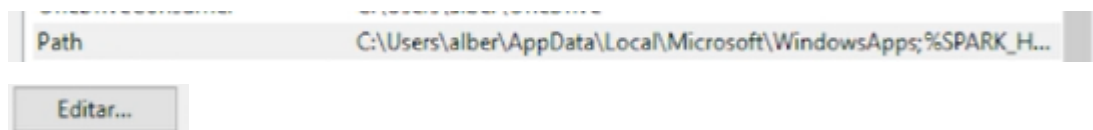
Para añadir las distintas variables:



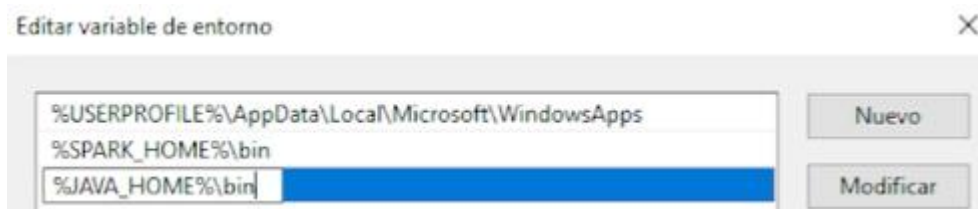
Tal que así:

SPARK_HOME	C:\spark\spark-3.5.1-bin-hadoop3
HADOOP_HOME	C:\spark\spark-3.5.1-bin-hadoop3
JAVA_HOME	C:\spark\jdk-17.0.11

En el path le damos a Editar



Y en “Nuevo” definimos %SPARK\_HOME%\bin y %JAVA\_HOME%\bin



#### 4) Comprobación de correcta instalación

Vamos al comand prompt



Nos situamos en la carpeta

```
C:\Users\alber>dir C:\spark
El volumen de la unidad C es Acer
El número de serie del volumen es: B82F-6625

Directorio de C:\spark

29/04/2024  21:33    <DIR>        .
29/04/2024  21:33    <DIR>        ..
11/03/2024  20:11    <DIR>        jdk-17.0.11
29/04/2024  21:28    <DIR>        spark-3.5.1-bin-hadoop3
                0 archivos             0 bytes
                4 dirs  659.941.773.312 bytes libres
```

Vamos a bin

```
C:\Users\alber>cd C:\spark\spark-3.5.1-bin-hadoop3\bin
```

Abrimos spark-shell

```
C:\spark\spark-3.5.1-bin-hadoop3\bin>spark-shell
```

Comprovamos

```
C:\spark\spark-3.5.1-bin-hadoop3>echo %JAVA_HOME%
C:\spark\jdk-17.0.11
```

Ejecutamos spark-shell

```
C:\spark\spark-3.5.1-bin-hadoop3\bin>spark-shell
```

#### 5) Dataset a utilizar



Abre un Terminal (en windows preferiblemente PowerShell) apuntando al directorio `../spark-x.y.z-bin-hadoop/bin/`

El dataset que utilizaremos se llama *Summer-Olympic-medals-1976-to-2008.csv*.

```
PowerShell 7.4.2
PS C:\Users\JoseD\Downloads\storage instaladores\PowerShell-7.4.2-win-x64> cd C:\spark\spark-3.5.1-bin-hadoop3\bin
PS C:\spark\spark-3.5.1-bin-hadoop3\bin> spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/05/01 19:23:17 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
Spark context Web UI available at http://JOSEANGULO:4040
Spark context available as 'sc' (master = local[*], app id = local-1714584199122).
Spark session available as 'spark'.
Welcome to

  ____  __
 / ___/  / /_  __
/ /   / __/ / / /
/ /   / /_ / /_/ /
/_/   /___/_/_/_/

version 3.5.1

Using Scala version 2.12.18 (Java HotSpot(TM) 64-Bit Server VM, Java 22.0.1)
Type in expressions to have them evaluated.
Type :help for more information.
```

### 3. Inspección de los datos locales

Carga el dataset de la siguiente manera (con la ruta del dataset que aplique):

```
val dataset = spark.read.option("header", "true").option("inferSchema",
"true").csv("/path/to/Summer-Olympic-medals-1976-to-2008.csv")
```

```
val olym_dataset =
spark.read.option("header", "true").option("inferSchema", "true").csv("C:/storage/Summer-
Olympic-medals-1976-to-2008.csv")
```

```
Using Scala version 2.12.18 (Java HotSpot(TM) 64-Bit Server VM, Java 22.0.1)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val olym_dataset = spark.read.option("header", "true").option("inferSchema", "true").csv("C:/storage/Summer-Olympic
-medals-1976-to-2008.csv")
olym_dataset: org.apache.spark.sql.DataFrame = [City: string, Year: int ... 9 more fields]

scala>
```

a) Explora el dataset

```
scala> val olym_dataset = spark.read.option("header", "true").option("inferSchema", "true").csv("C:/storage/Summer-Olympic-medals-1976")
oly_dataset: org.apache.spark.sql.DataFrame = [City: string, Year: int ... 9 more fields]

scala> olym_dataset.show()
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| City|Year| Sport|Discipline| Event| Athlete|Gender|Country_Code| Country|Event_gender| Medal|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|Montreal|1976|Aquatics| Diving| 3m springboard| K?HLER, Christa| Women| GDR| East Germany| W| Silver|
|Montreal|1976|Aquatics| Diving| 3m springboard| KOSENKOV, Aleksandr| Men| URS| Soviet Union| M| Bronze|
|Montreal|1976|Aquatics| Diving| 3m springboard| BOGGS, Philip George| Men| USA| United States| M| Gold|
|Montreal|1976|Aquatics| Diving| 3m springboard| CAGNOTTO, Giorgio...| Men| ITA| Italy| M| Silver|
|Montreal|1976|Aquatics| Diving| 10m platform| WILSON, Deborah K...| Women| USA| United States| W| Bronze|
|Montreal|1976|Aquatics| Diving| 10m platform| LOUGANIS, Gregory| Men| USA| United States| M| Silver|
|Montreal|1976|Aquatics| Diving| 10m platform| VAYTSEKHOVSKAYA, ...| Women| URS| Soviet Union| W| Gold|
|Montreal|1976|Aquatics| Diving| 3m springboard| POTTER-MCINGVALE, ...| Women| USA| United States| W| Bronze|
|Montreal|1976|Aquatics| Diving| 10m platform| DIBIASI, Klaus| Men| ITA| Italy| M| Gold|
|Montreal|1976|Aquatics| Diving| 10m platform| ALEINIK, Vladimir| Men| URS| Soviet Union| M| Bronze|
|Montreal|1976|Aquatics| Diving| 10m platform| KNAPE-LINDBERGH, ...| Women| SWE| Sweden| W| Silver|
|Montreal|1976|Aquatics| Diving| 3m springboard| CHANDLER, Jennife...| Women| USA| United States| W| Gold|
|Montreal|1976|Aquatics| Swimming| 4x100m freestyle ...| BABASHOFF, Shirle...| Women| USA| United States| W| Gold|
|Montreal|1976|Aquatics| Swimming| 400m freestyle| SHAW, Timothy Andrew| Men| USA| United States| W| Gold|
```

b) ¿Que han aportado las opciones “header” y “inferSchema”?

`oly_dataset.head()`

Este comando permite ver todos los datos de la primera fila del dataset explorado sin el nombre de las columnas.

`oly_dataset.schema`

Permite ver las características que tiene cada columna del dataset como el tipo de dato.

```

|Montreal|1976|Aquatics| Swimming| 800m freestyle| TH?MER, Petra| Women| GDR| East Germany| W| Gold|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 20 rows

scala> oly_dataset.head()
res1: org.apache.spark.sql.Row = [Montreal,1976,Aquatics,Diving,3m springboard,K?HLER, Christa,Women,GDR,East Germany,W, Silver]

scala> oly_dataset.schema
res2: org.apache.spark.sql.types.StructType = StructType(StructField(City,StringType,true),StructField(Year,IntegerType,true),StructField(Sport,StringType,true),StructField(Discipline,StringType,true),StructField(Event,StringType,true),StructField(Athlete,StringType,true),StructField(Gender,StringType,true),StructField(Country_Code,StringType,true),StructField(Country,StringType,true),StructField(Event_gender,StringType,true),StructField(Medal,StringType,true))

scala>
```



c) ¿Como harías para contar las medallas conseguidas por año y país?

Primero importar la librería de funciones de sql para que el spark pueda interpretar las sentencias que se le van a entregar a través de una variable, donde se agrupara por año y país, contando las medallas que tiene cada uno.

```
scala> import org.apache.spark.sql.functions._
import org.apache.spark.sql.functions._
```

```
scala> val totalmedallas = oym_dataset.groupBy("Year", "Country").agg(count("Medal"))
totalmedallas: org.apache.spark.sql.DataFrame = [Year: int, Country: string ... 1 more field]
```

```
scala> totalmedallas.show
+-----+-----+-----+
|Year|      Country|count(Medal)|
+-----+-----+-----+
|1996|      Romania|          38|
|2000|       Russia|         188|
|2004|        India|           1|
|2008|     Portugal|           2|
|2008|    Mauritius|           1|
|1980|      Finland|           9|
|1996|      Austria|           3|
|1988|   Switzerland|           8|
|1992|         China|          83|
|1996|      Ukraine|          34|
|2000|      Algeria|           5|
|2008|   Netherlands|          62|
|1980|       Guyana|           1|
|1988| Soviet Union|         294|
|1976|      Pakistan|          16|
|1976| Korea, South|          17|
|1976|      Jamaica|           2|
|1980| East Germany|         260|
|1984|      Nigeria|           5|
|1992| Czechoslovakia|          8|
+-----+-----+-----+
only showing top 20 rows

scala>
```

d) Usando SparkSQL muestra alguna métrica interesante.

Para poder ejecutar alguna sentencia select, se debe crear una vista de la variable que contiene el dataset. Y luego crear otra variable que contenta la consulta a realizar. Como sigue:

```
oym_dataset.createTempView("agrupacion")
```

```
val cantciudad = spark.sql("""SELECT City, Sport, COUNT(*) AS CantMedallas FROM agrupacion
GROUP BY City, Sport ORDER BY CantMedallas DESC""")
```

```

scala> olym_dataset.createTempView("agrupacion")

scala> val cantciudad = spark.sql("""SELECT City,Sport,COUNT(*) AS CantMedallas FROM agrupacion GROUP BY City,Sport ORDER BY CantMedallas DESC""")
cantciudad: org.apache.spark.sql.DataFrame = [City: string, Sport: string ... 1 more field]

scala> cantciudad.show
+-----+
| City | Sport | CantMedallas |
+-----+
| Beijing | Aquatics | 347 |
| Athens | Aquatics | 332 |
| Sydney | Aquatics | 329 |
| Atlanta | Aquatics | 262 |
| Barcelona | Aquatics | 228 |
| Seoul | Aquatics | 202 |
| Los Angeles | Aquatics | 192 |
| Sydney | Athletics | 184 |
| Athens | Athletics | 183 |
| Atlanta | Athletics | 180 |
| Barcelona | Athletics | 178 |
| Beijing | Athletics | 177 |
| Seoul | Athletics | 163 |
| Moscow | Rowing | 162 |
| Los Angeles | Rowing | 162 |
| Montreal | Rowing | 162 |
| Los Angeles | Athletics | 161 |
| Seoul | Rowing | 159 |
| Moscow | Aquatics | 159 |
| Montreal | Aquatics | 159 |
+-----+
only showing top 20 rows

```

e) ¿Que muestra la SparkUI: <http://localhost:4040> ?

SPARK\_UI LOCAL: <http://joseangulo:4040/jobs/>

Observar que cada imagen es la vista de las opciones de arriba de esta pagina: JOBS, STAGES, ENVIROMENT, EXECUTORS, SQL

**VISTA JOBS:** Los Trabajos que se han ejecutado con sus características principales de performance.

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
7	show at <console>27 show at <console>27	2024/05/01 20:28:38	0.1 s	1/1 (1 skipped)	1/1 (1 skipped)
6	show at <console>27 show at <console>27	2024/05/01 20:28:37	1 s	1/1	1/1
5	show at <console>27 show at <console>27	2024/05/01 20:03:47	0.1 s	1/1 (1 skipped)	1/1 (1 skipped)
4	show at <console>27 show at <console>27	2024/05/01 20:03:45	2 s	1/1	1/1
3	head at <console>24 head at <console>24	2024/05/01 19:48:07	27 ms	1/1	1/1
2	show at <console>24 show at <console>24	2024/05/01 19:36:12	0.2 s	1/1	1/1
1	csv at <console>22 csv at <console>22	2024/05/01 19:34:18	0.3 s	1/1	1/1
0	csv at <console>22 csv at <console>22	2024/05/01 19:34:17	0.5 s	1/1	1/1

VISTA DE STAGES:

Spark shell - Stages for All Jobs

Jose Angulo

Stages for All Jobs

Completed Stages: 8  
Skipped Stages: 2

Completed Stages (8)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
9	show at <console>:27	2024/05/01 20:28:38	0,1 s	1/1			15,7 KB	
7	show at <console>:27	2024/05/01 20:28:37	1 s	1/1	1422,9 KB			15,7 KB
6	show at <console>:27	2024/05/01 20:03:47	0,1 s	1/1			24,7 KB	
4	show at <console>:27	2024/05/01 20:03:45	2 s	1/1	1422,9 KB			24,7 KB
3	head at <console>:24	2024/05/01 19:48:07	27 ms	1/1	64,0 KB			
2	show at <console>:24	2024/05/01 19:36:12	0,1 s	1/1	64,0 KB			
1	csv at <console>:22	2024/05/01 19:34:18	0,3 s	1/1	1422,9 KB			
0	csv at <console>:22	2024/05/01 19:34:18	0,4 s	1/1	64,0 KB			

Skipped Stages (2)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
8	show at <console>:27	Unknown	Unknown	0/1				
5	show at <console>:27	Unknown	Unknown	0/1				

VISTA DE LOS ENVIRONMENT:

Spark shell - Environment

Jose Angulo

Environment

Runtime Information

Name	Value
Java Home	C:\spark\java
Java Version	22.0.1 (Oracle Corporation)
Scala Version	version 2.12.18

Spark Properties

Name	Value
spark.app.id	local-1714584199122
spark.app.name	Spark shell
spark.app.startTime	1714584197562
spark.app.submitTime	1714584191342
spark.driver.extraJavaOptions	-Djava.net.preferIPv6Addresses=false -XX:IgnoreUnrecognizedVMOptions --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.lang.invoke=ALL-UNNAMED --add-opens=java.base/java.lang.reflect=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.base/java.net=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.util.concurrent=ALL-UNNAMED --add-opens=java.base/java.util.concurrent.atomic=ALL-UNNAMED --add-opens=java.base/jdk.internal.ref=ALL-UNNAMED --add-opens=java.base/sun.nio.ch=ALL-UNNAMED --add-opens=java.base/sun.nio.cs=ALL-UNNAMED --add-opens=java.base/sun.security.action=ALL-UNNAMED --add-opens=java.base/sun.util.calendar=ALL-UNNAMED --add-opens=java.security.jgss/sun.security.krb5=ALL-UNNAMED -Djdk.reflect.useDirectMethodHandle=false
spark.driver.host	JOSEANGULO
spark.driver.port	49918
spark.executor.extraJavaOptions	-Djava.net.preferIPv6Addresses=false -XX:IgnoreUnrecognizedVMOptions --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.lang.invoke=ALL-UNNAMED --add-opens=java.base/java.lang.reflect=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.base/java.net=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED

VISTA DE LOS EXECUTORS:

## Executors

[Show Additional Metrics](#)

### Summary

	+	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Excluded
Active(t)	0	20.4 KIB / 434.4 MIB	0.0 B	16	0	0	0	8	8	1.2 h (0.5 s)	4.4 MIB	40.4 KIB	40.4 KIB	0
Dead(t)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0
Total(t)	0	20.4 KIB / 434.4 MIB	0.0 B	16	0	0	0	8	8	1.2 h (0.5 s)	4.4 MIB	40.4 KIB	40.4 KIB	0

### Executors

Show 20 entries

Search:

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Thread Dump	Heap Histogram	Add Time	Remove Time
driver	JOSEANGULO@9919	Active	0	20.4 KIB / 434.4 MIB	0.0 B	16	0	0	8	8	1.2 h (0.5 s)	4.4 MIB	40.4 KIB	40.4 KIB	<a href="#">Thread Dump</a>	<a href="#">Heap Histogram</a>	2024-05-01 19:23:19	-

Showing 1 to 1 of 1 entries

Previous 1 Next

## VISTA DE SQL:

## SQL / DataFrame

Completed Queries: 6

[Completed Queries \(6\)](#)

Page: 1

1 Pages. Jump to 1. Show 100 items in a page. Go

ID	Description	Submitted	Duration	Job IDs
5	<a href="#">show at &lt;console&gt;:27</a>	2024/05/01 20:28:37	2 s	<a href="#">[6][7]</a>
4	<a href="#">createTempView at &lt;console&gt;:27</a>	2024/05/01 20:26:05	10 ms	
3	<a href="#">show at &lt;console&gt;:27</a>	2024/05/01 20:03:45	2 s	<a href="#">[4][5]</a>
2	<a href="#">head at &lt;console&gt;:24</a>	2024/05/01 19:48:07	98 ms	<a href="#">[3]</a>
1	<a href="#">show at &lt;console&gt;:24</a>	2024/05/01 19:36:12	0.3 s	<a href="#">[2]</a>
0	<a href="#">csv at &lt;console&gt;:22</a> csv at <console>:22 org.apache.spark.sql.DataFrameReader.csv(DataFrameReader, scala:444) \$line14.\$read\$\$in\$10\$line14\$line14\$line14\$.cinit(<console>:22) \$line14.\$read\$\$in\$10\$line14\$line14\$line14\$.cinit(<console>:26) \$line14.\$read\$\$in\$10\$line14\$line14\$line14\$.cinit(<console>:28) \$line14.\$read\$\$in\$10\$line14\$line14\$line14\$.cinit(<console>:30) \$line14.\$read\$\$in\$10\$line14\$line14\$line14\$.cinit(<console>:32) \$line14.\$read\$\$in\$10\$line14\$.cinit(<console>:34) \$line14.\$read\$\$in\$.cinit(<console>:36) \$line14.\$read\$.cinit(<console>:38) \$line14.\$read\$.cinit(<console>:40) \$line14.\$read\$.cinit(<console>:44) \$line14.\$read\$.cinit(<console>:44)	2024/05/01 19:34:17	1 s	<a href="#">[0]</a>