Unidad 2: Modelo de mapeo y reducción

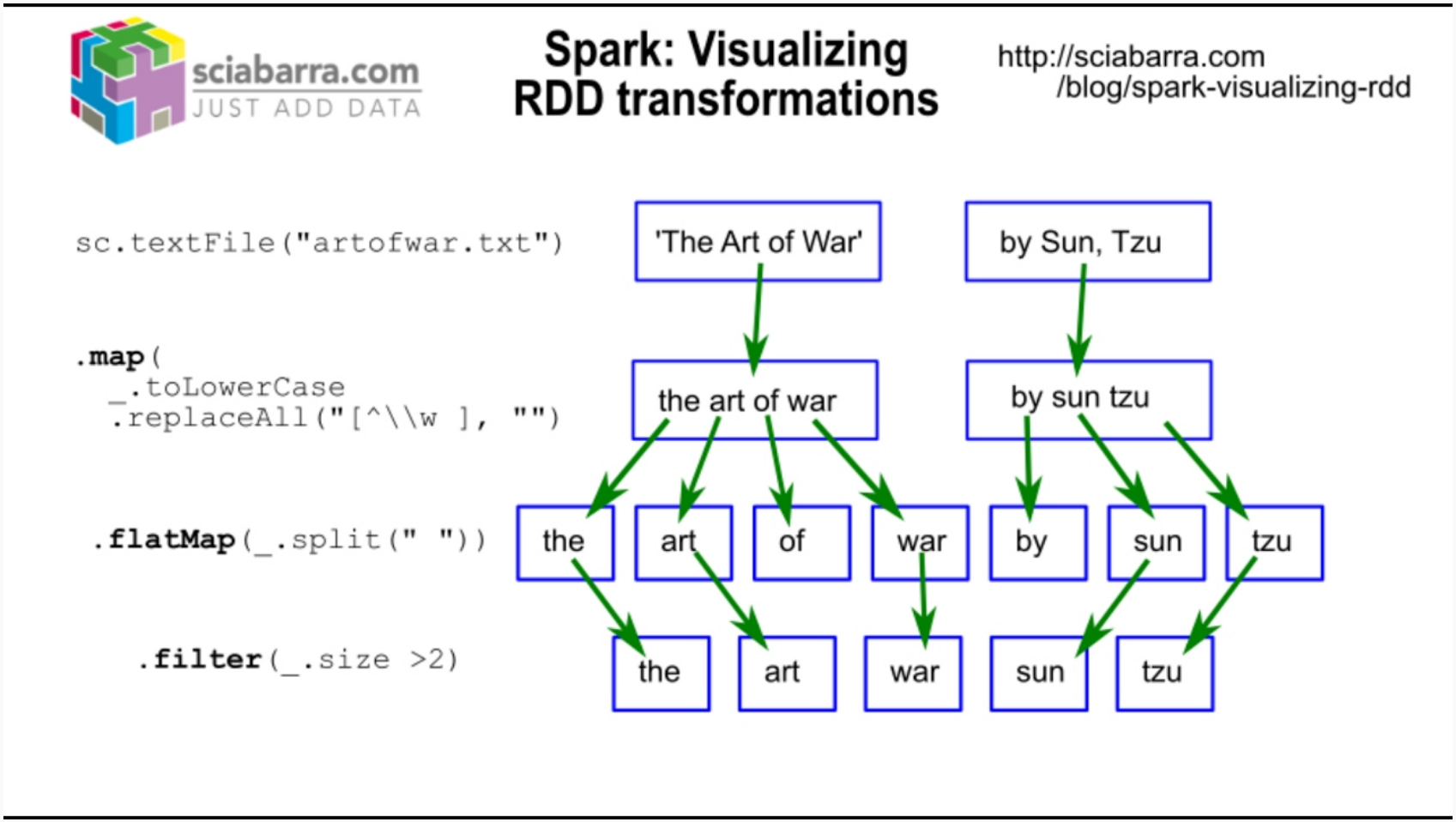
Práctica: Apache Spark

Blanca Vázquez

Febrero 2020

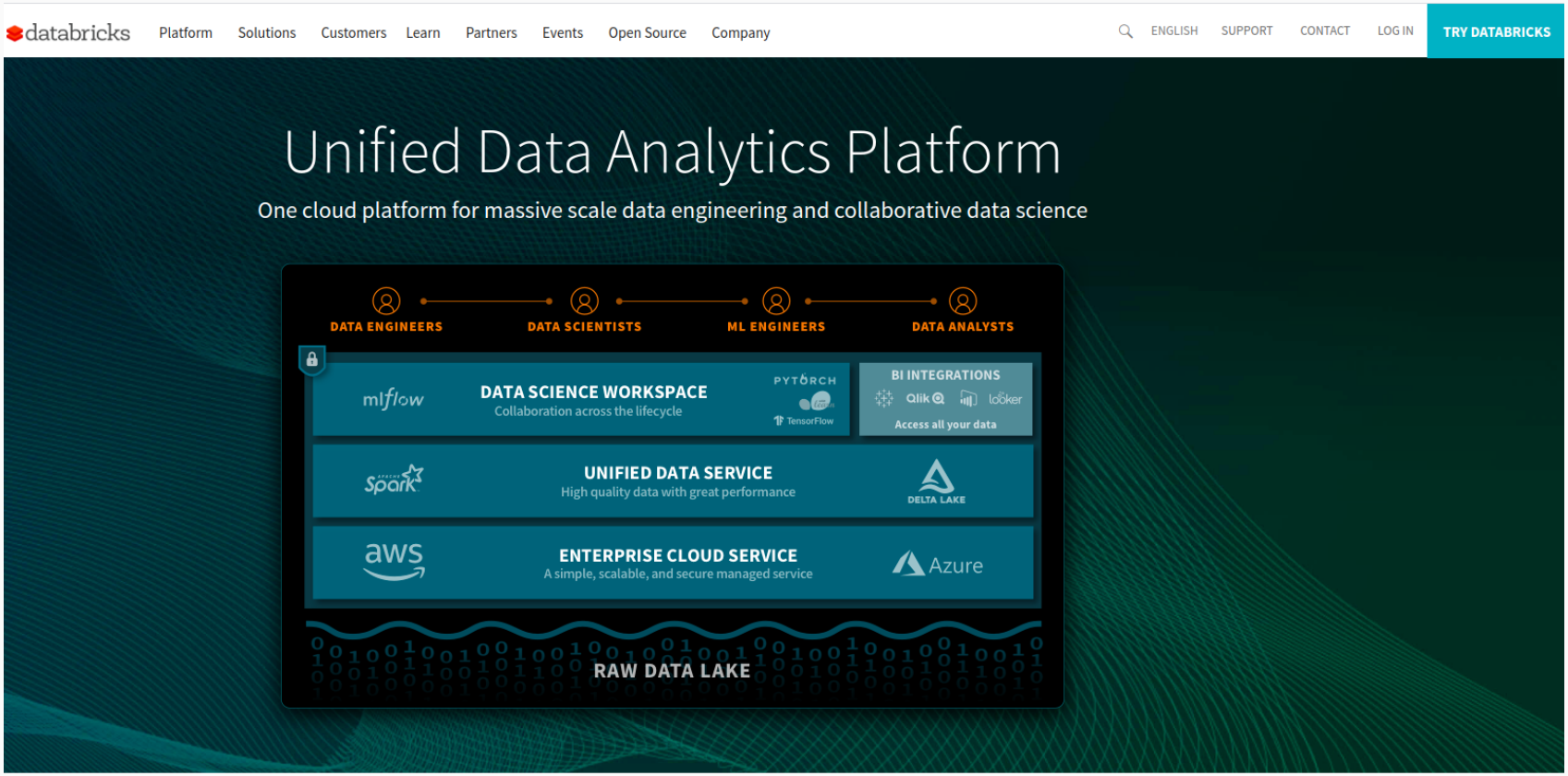
Es un framework de computación basado en clúster open-source.

- Se caracteriza por su velocidad para el procesamiento de big data y aprendizaje automatizado.
- Utiliza datos distribuidos resilentes (RDD): es una colección de elementos particionados en los nodos del clúster para ser operados en paralelo.


sciabarra.com
JUST ADD DATA
Spark: Visualizing RDD transformations
http://sciabarra.com /blog/spark-visualizing-rdd
sc.textFile("artofwar.txt")
.map(
_.toLowerCase
.replaceAll("[^\\w], "")
.flatMap(_.split(" "))
.filter(_.size >2)
'The Art of War'
by Sun, Tzu
the art of war
by sun tzu
the
art
of
war
by
sun
tzu
the
art
war
sun
tzu

Databricks

Es una plataforma de análisis unificada, de los creadores de Apache Spark. Facilita la creación de clúster Spark en la nube.



https://databricks.com/

Registro

DATABRICKS PLATFORM – FREE TRIAL

For businesses looking for a zero-management cloud platform built around Apache Spark

- Unlimited clusters that can scale to any size
- Job scheduler to execute jobs for production pipelines
- Fully interactive notebook with collaboration, dashboards, REST APIs
- Advanced security, role-based access controls, and audit logs
- Single Sign On support
- Integration with BI tools such as Tableau, Qlik, and Looker
- 14-day full feature trial (excludes cloud charges)

GET STARTED

COMMUNITY EDITION

For students and educational institutions just getting started with Apache Spark

- Single cluster limited to 6GB and no worker nodes
- Basic notebook without collaboration
- Limited to 3 max users
- Public environment to share your work

GET STARTED

Registro

Sign Up for Databricks Community Edition

First Name *

Last Name *

Company Name *

Work Email *

Phone Number

What is your intended use case? *

- Please Select -

How would you describe your role? *

- Please Select -

☑ Keep me informed with the occasional update about Databricks and Apache Spark™.

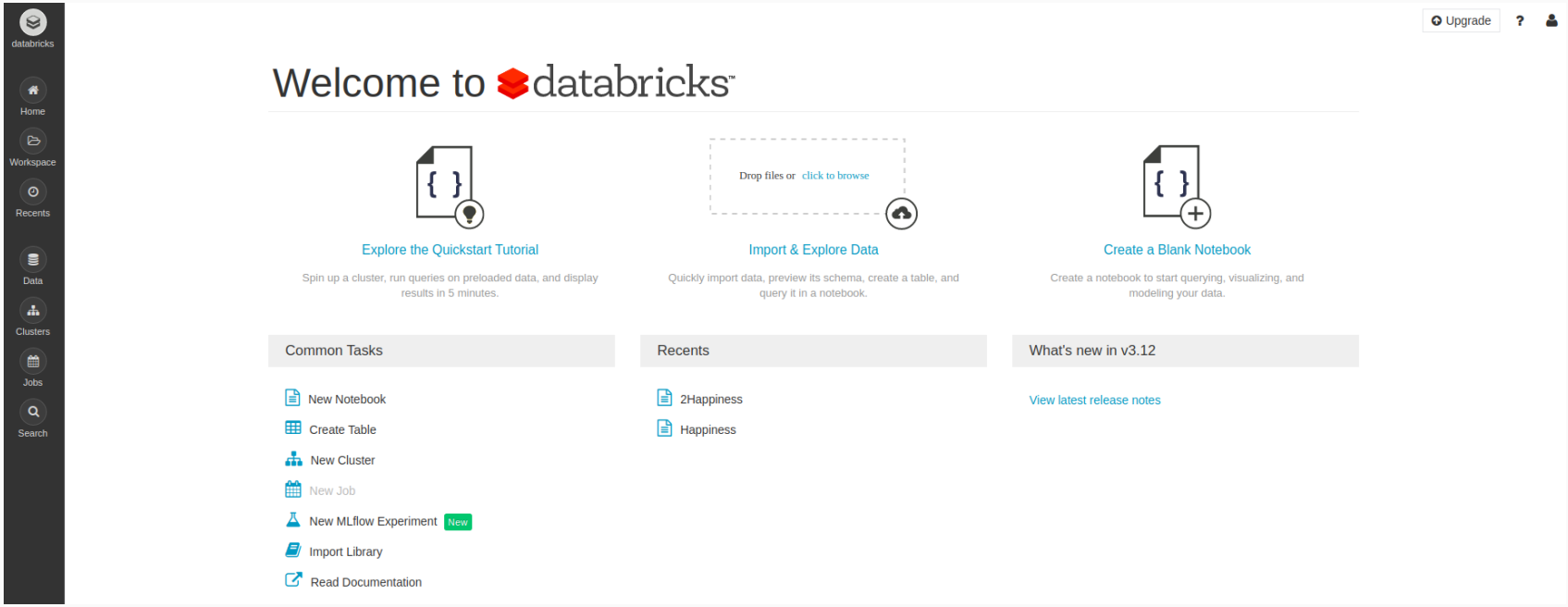
By clicking "Sign Up", you agree to the Terms of Service and the Privacy Policy.

☐ No soy un robot

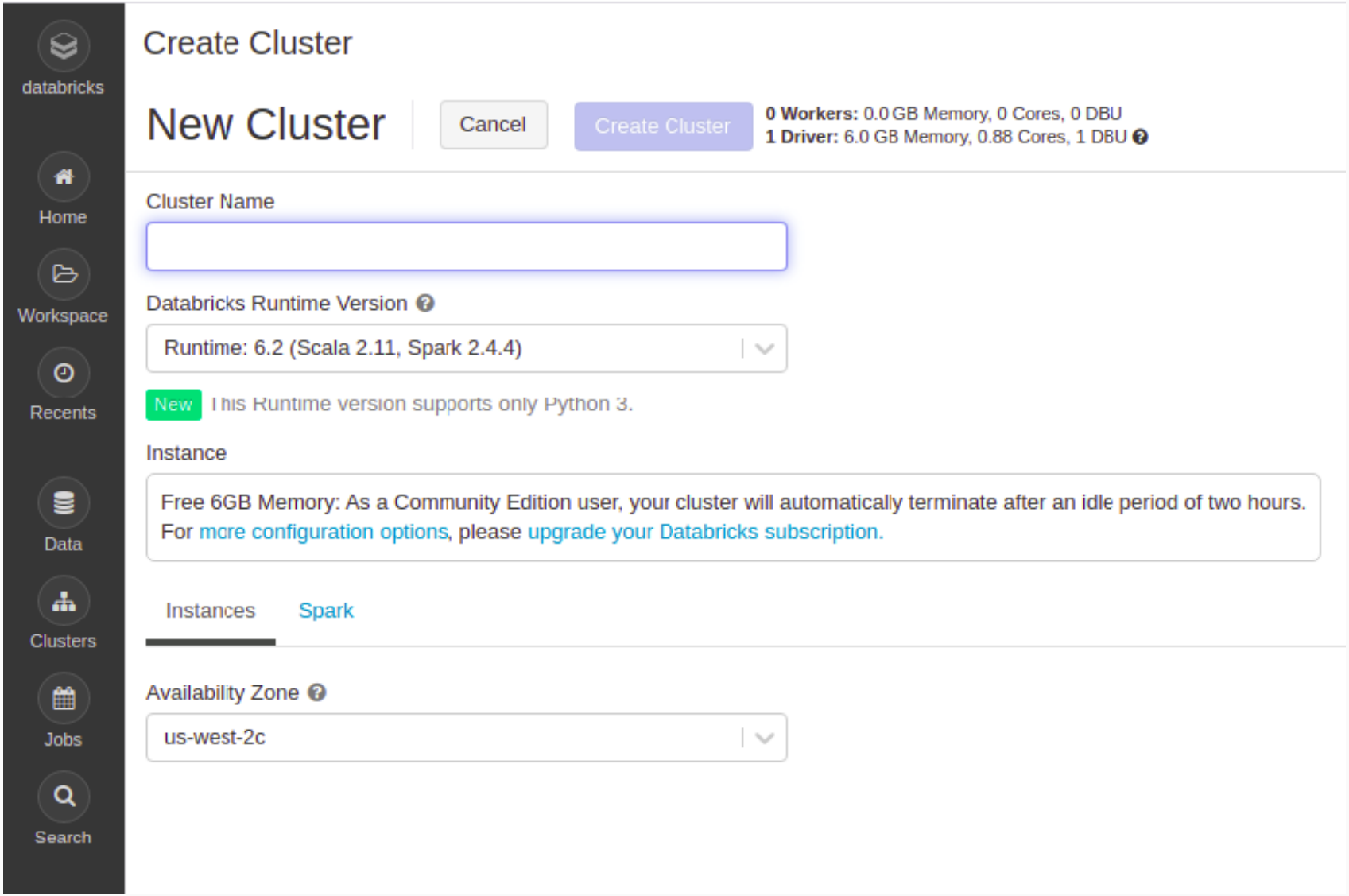
reCAPTCHA
Privacidad - Condiciones

Sign Up

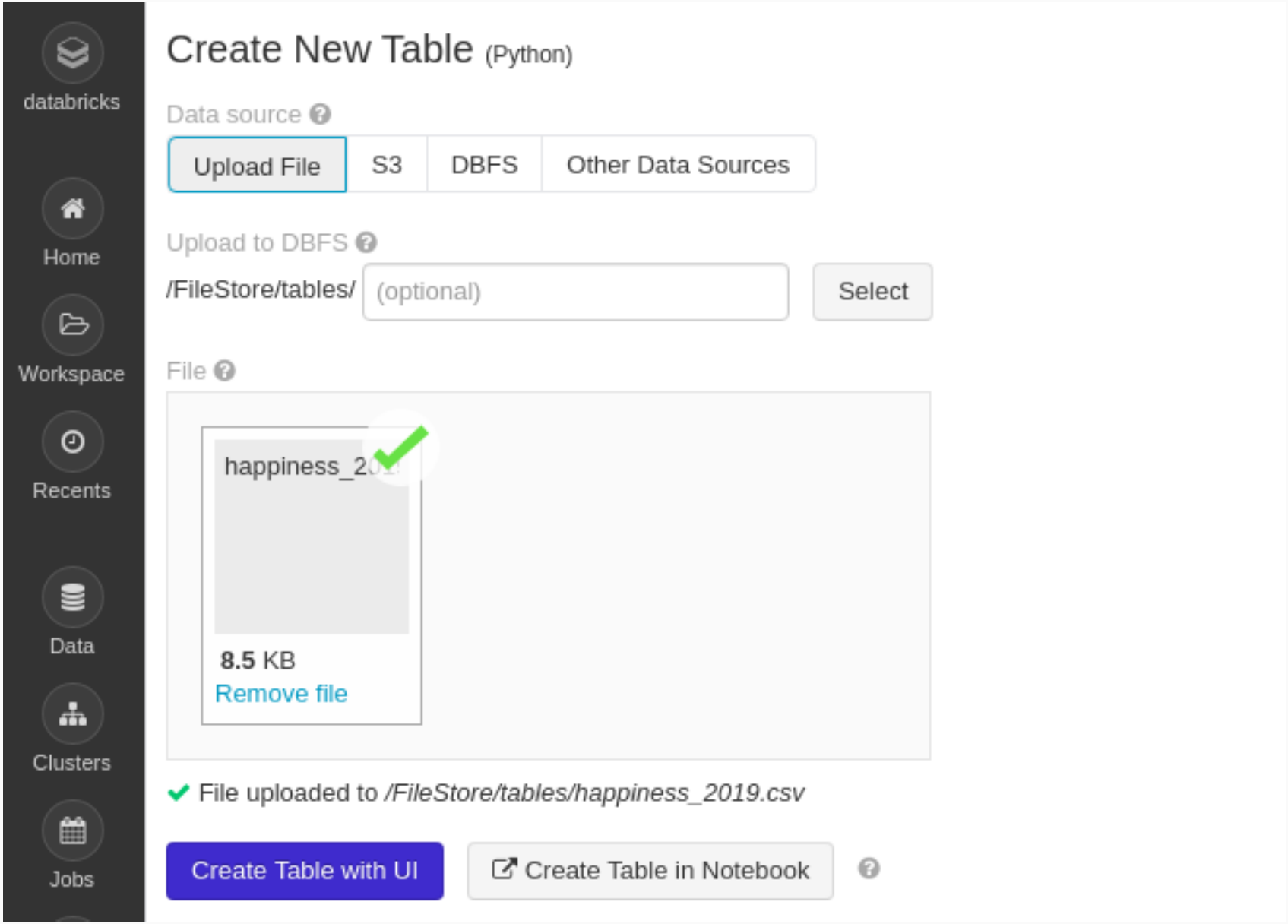
Página principal de Databricks



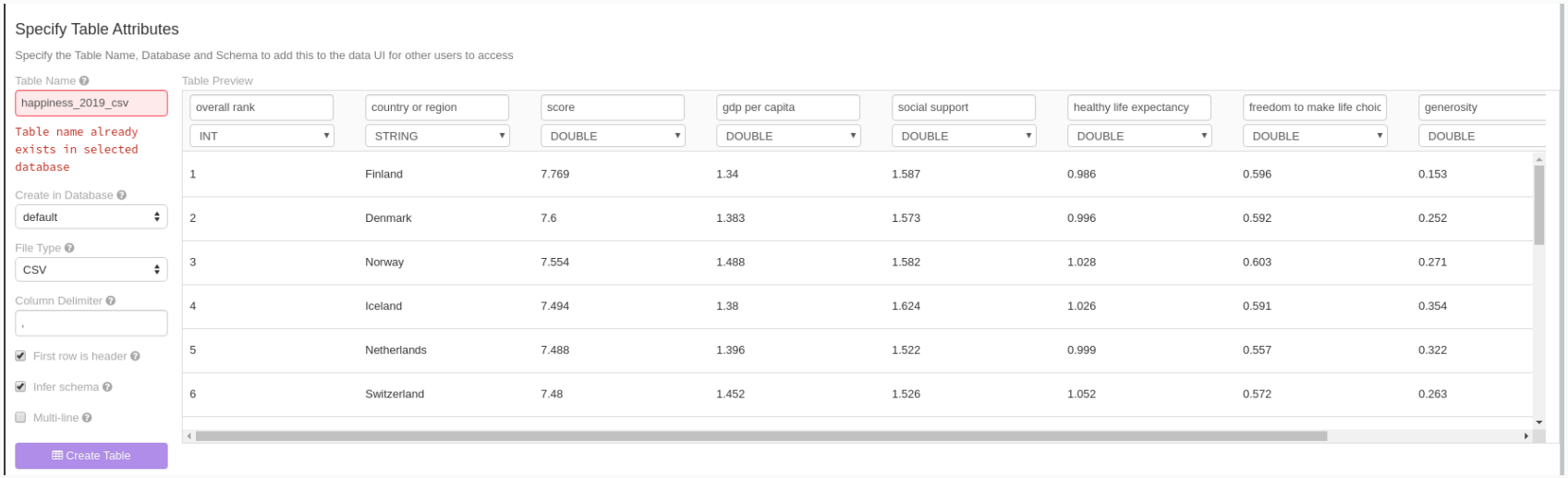
Creación de clúster



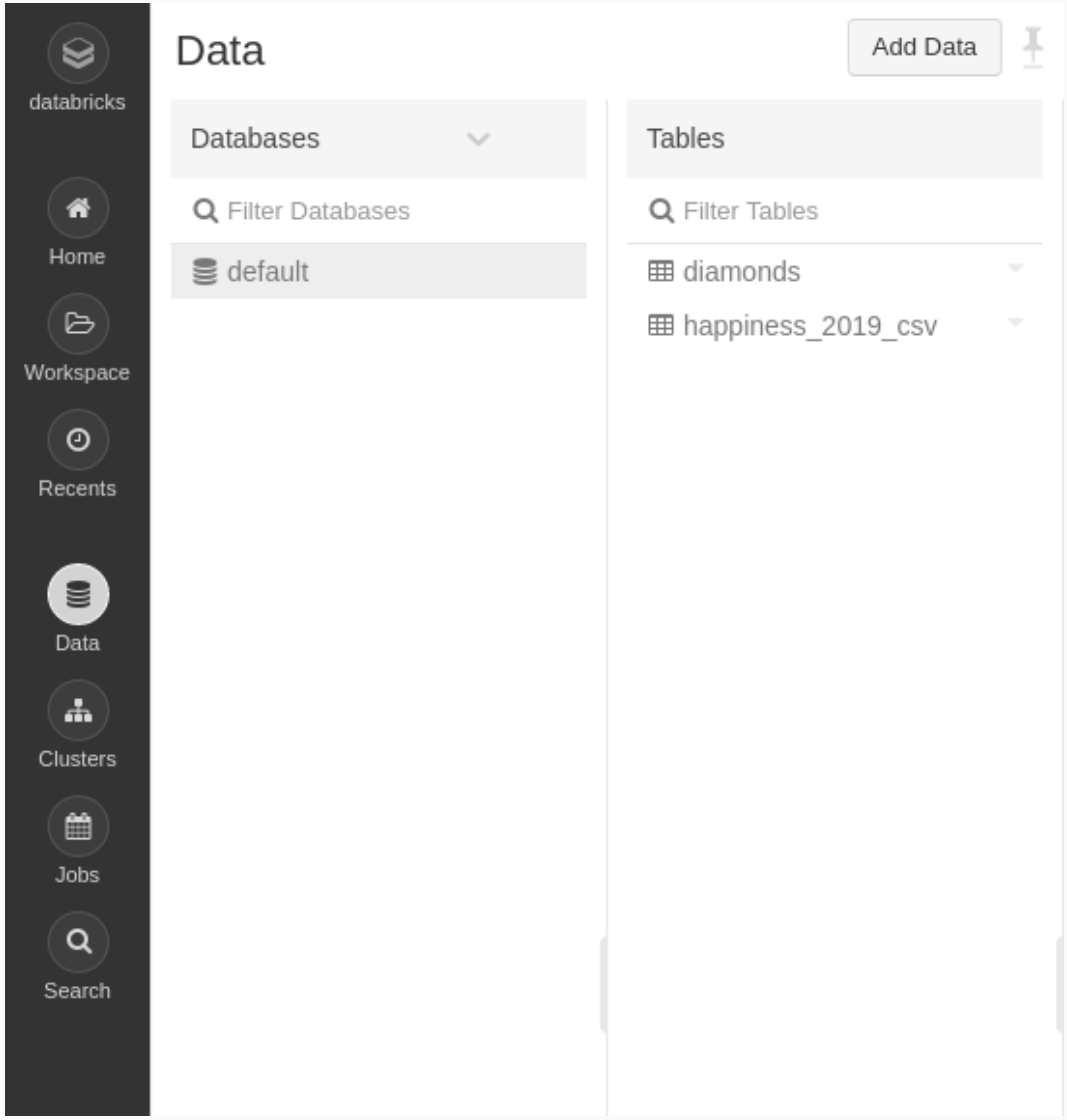
Cargar datos



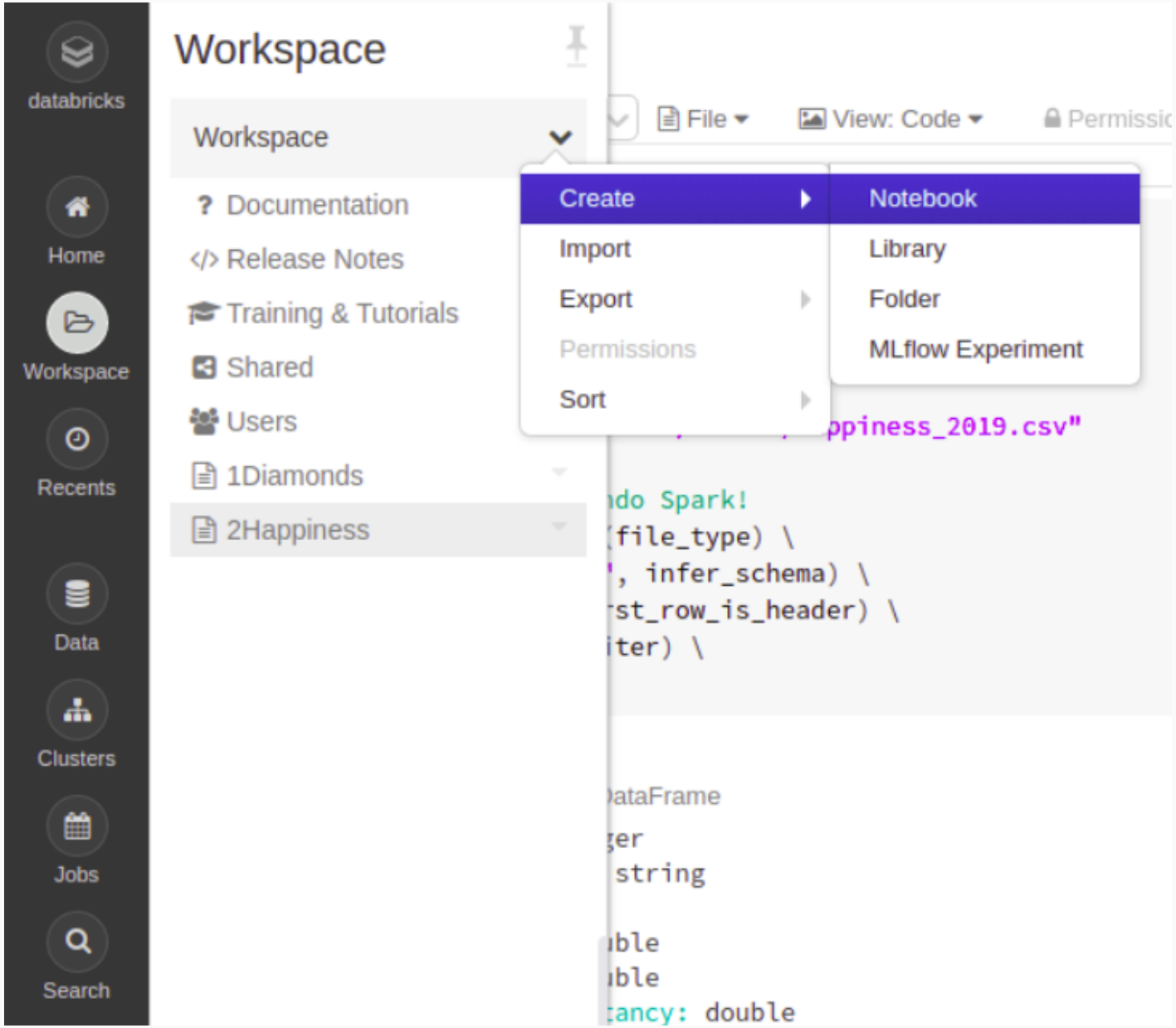
Cargar datos



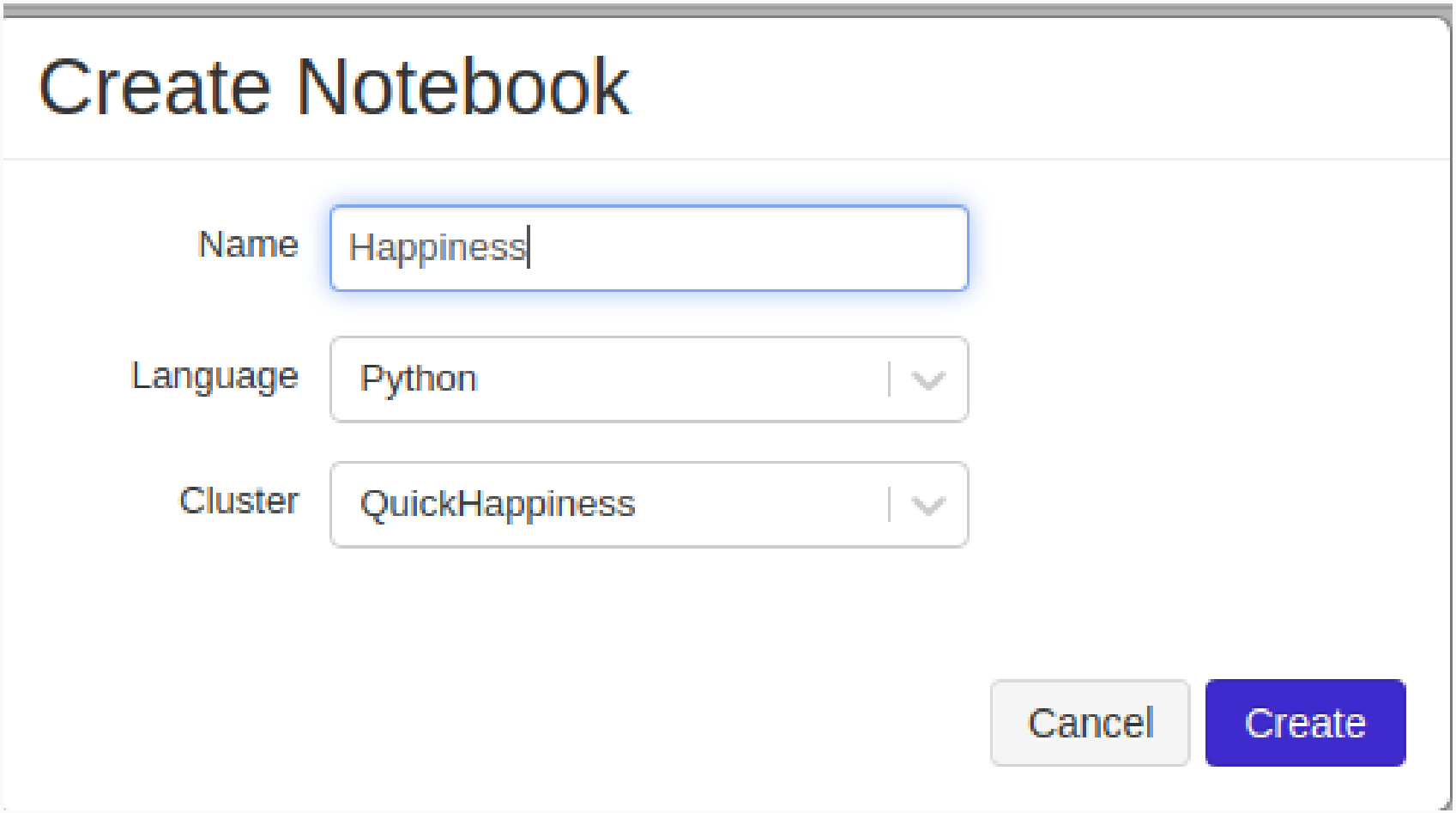
Cargar datos



Crear cuaderno de trabajo



Crear cuaderno de trabajo



Lectura de datos usando Apache Spark

2Happiness (Python)

QuickHappiness

Cmd 1

```python
#Opciones CSV
file_type = "csv"
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","
file_location = "/FileStore/tables/happiness_2019.csv"

# Lectura de datos usando Spark!
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)
```

(2) Spark Jobs

df: pyspark.sql.dataframe.DataFrame

```
overall rank: integer
country or region: string
score: double
gdp per capita: double
social support: double
healthy life expectancy: double
freedom to make life choices: double
generosity: double
perceptions of corruption: double
```

Command took 0.96 seconds -- by blancavazquez@comunidad.unam.mx at 2/5/2020, 8:15:09 AM on QuickHappiness

Visualizando registros en tablas

display(df.head(10))

Cmd 2

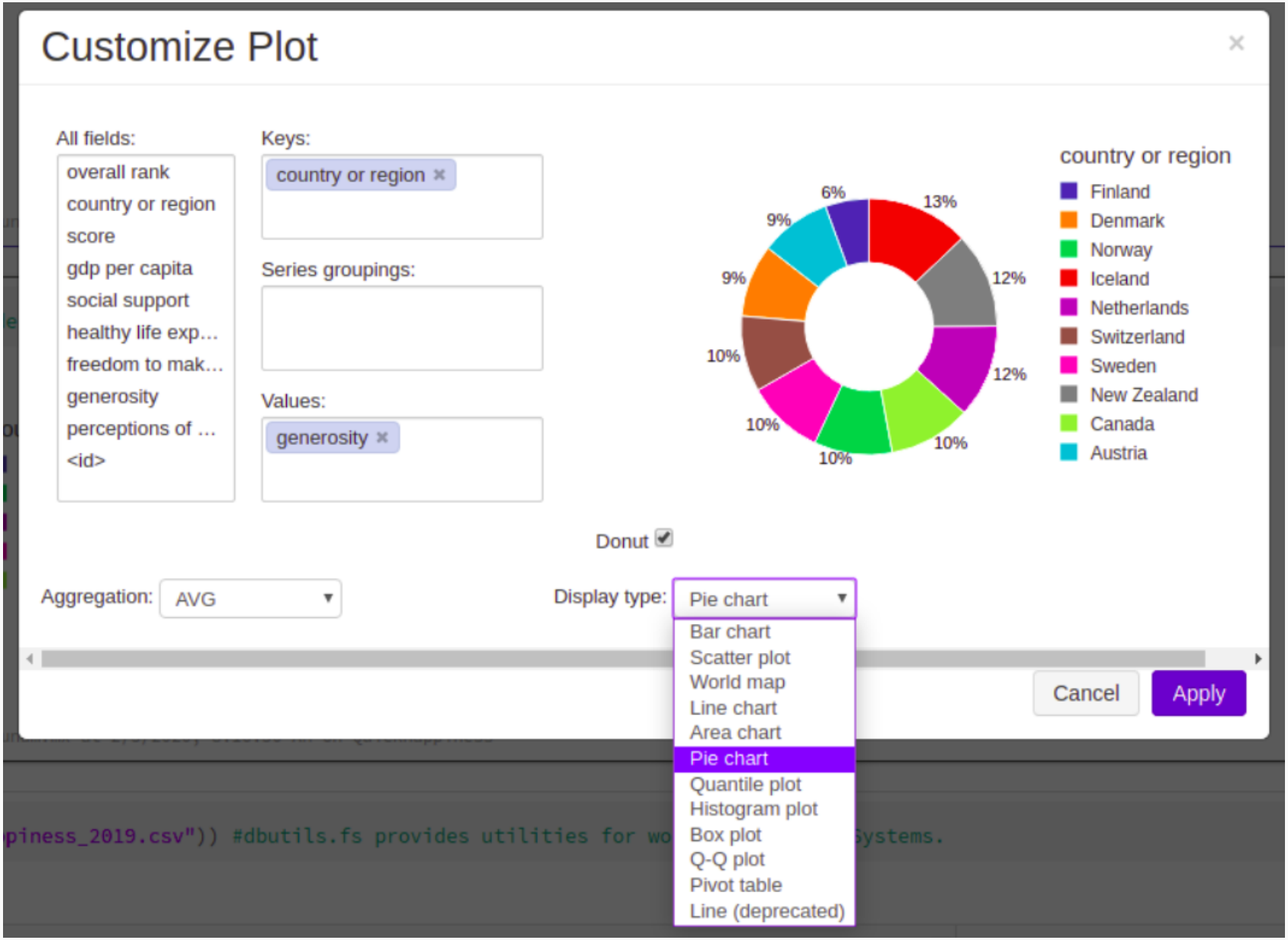
```
1 display(df.head(10)) #Muestra el contenido de los primeros 10 registros de la tabla happiness_2019.csv
```

(4) Spark Jobs

overall rank	country or region	score	gdp per capita	social support	healthy life expectancy	freedom to make life choices	generosity	perceptions of corruption
1	Finland	7.769	1.34	1.587	0.986	0.596	0.153	0.393
2	Denmark	7.6	1.383	1.573	0.996	0.592	0.252	0.41
3	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341
4	Iceland	7.494	1.38	1.624	1.026	0.591	0.354	0.118
5	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298
6	Switzerland	7.48	1.452	1.526	1.052	0.572	0.263	0.343
7	Sweden	7.343	1.387	1.487	1.009	0.574	0.267	0.373
8	New Zealand	7.307	1.303	1.557	1.026	0.585	0.33	0.38
9	Canada	7.278	1.365	1.505	1.039	0.584	0.285	0.308

Command took 1.65 seconds -- by blancavazquez@comunidad.unam.mx at 2/5/2020, 8:16:30 AM on QuickHappiness

Visualizando datos en forma de gráficas



Visualizando información de los datos

Cmd 3

```
1 display(dbutils.fs.ls("/FileStore/tables/happiness_2019.csv")) #dbutils.fs provides utilities for working with FileSystems.
```

(3) Spark Jobs

path	name	size
dbfs:/FileStore/tables/happiness_2019.csv	happiness_2019.csv	8510

Command took 0.62 seconds -- by blancavazquez@comunidad.unam.mx at 2/5/2020, 8:22:55 AM on QuickHappiness

Datos distribuidos resilentes (RDD)

Cmd 4

```
#Cargamos el archivo hacia un Resilient Distributed Dataset(RDD)
data_file = "/FileStore/tables/happiness_2019.csv"
raw_rdd = sc.textFile(data_file).cache()
raw_rdd.take(5) #show the top 5 lines of the file
```

(1) Spark Jobs

```
Out[5]: ['overall rank,country or region,score,gdp per capita,social support,healthy life expectancy,freedom to make life choices,generosity,per
ceptions of corruption',
 '1,Finland,7.769,1.34,1.587,0.986,0.596,0.153,0.393',
 '2,Denmark,7.6,1.383,1.573,0.996,0.592,0.252,0.41',
 '3,Norway,7.554,1.488,1.582,1.028,0.603,0.271,0.341',
 '4,Iceland,7.494,1.38,1.624,1.026,0.591,0.354,0.118']
```

Command took 0.26 seconds -- by blancavazquez@comunidad.unam.mx at 2/5/2020, 8:23:30 AM on QuickHappiness

Validación de la creación de una clase tipo RDD

Cmd 5

```
1 type(raw_rdd)#se imprime el tipo de clase del argumento pasado como parámetro
```

```
Out[6]: pyspark.rdd.RDD
```

Command took 0.03 seconds -- by blancavazquez@comunidad.unam.mx at 2/5/2020, 8:23:38 AM on QuickHappiness

Preparando RDD

Cmd 6

```
#En RDD, es necesario separar cada una de las entradas, antes de parsear y construir un dataframe.
csv_rdd = raw_rdd.map(lambda row: row.split(","))
print(csv_rdd.take(3))#print 2 rows
```

(1) Spark Jobs

```
[['overall rank', 'country or region', 'score', 'gdp per capita', 'social support', 'healthy life expectancy', 'freedom to make life choices',
'generosity', 'perceptions of corruption'], ['1', 'Finland', '7.769', '1.34', '1.587', '0.986', '0.596', '0.153', '0.393'], ['2', 'Denmark', '7.
6', '1.383', '1.573', '0.996', '0.592', '0.252', '0.41']]
```

Command took 0.17 seconds -- by blancavazquez@comunidad.unam.mx at 2/5/2020, 8:57:09 AM on QuickHappiness

Cmd 7

```
print(type(csv_rdd))#print types
```

```
<class 'pyspark.rdd.PipelinedRDD'>
```

Command took 0.02 seconds -- by blancavazquez@comunidad.unam.mx at 2/5/2020, 8:44:49 AM on QuickHappiness

Cmd 8

```
print('No. de columnas: ', len(csv_rdd.take(1)[0]))
```

(1) Spark Jobs

```
No. de columnas:  9
```

Command took 0.15 seconds -- by blancavazquez@comunidad.unam.mx at 2/5/2020, 8:45:27 AM on QuickHappiness

Preparando RDD

PipelinedRDD: representa una colección inmutable y particionada de elementos que pueden operar en paralelo. Es una subclase de RDD.

Home Trees Indices Help PySpark

Package pyspark :: Module rdd :: Class RDD [frames] | no frames

Class RDD

source code

```
object --+
         |
        RDD
```

Known Subclasses:
PipelinedRDD

A Resilient Distributed Dataset (RDD), the basic abstraction in Spark. Represents an immutable, partitioned collection of elements that can be operated on in parallel.

Instance Methods		
	`__init__(self, jrdd, ctx)` x.__init__(...) initializes x; see help(type(x)) for signature	source code
	`cache(self)` Persist this RDD with the default storage level (`MEMORY_ONLY`).	source code
	`persist(self, storageLevel)` Set this RDD's storage level to persist its values across operations after the first time it is computed.	source code
	`unpersist(self)` Mark the RDD as non-persistent, and remove all blocks for it from memory and disk.	source code
	`checkpoint(self)` Mark this RDD for checkpointing.	source code
	`isCheckpointed(self)` Return whether this RDD has been checkpointed or not	source code
	`getCheckpointFile(self)` Gets the name of the file to which this RDD was checkpointed	source code
	`map(self, f, preservesPartitioning=False)` Return a new RDD containing the distinct elements in this RDD.	source code

https://spark.apache.org/docs/0.8.0/api/pyspark/pyspark.rdd.RDD-class.html

Creación de tabla usando pyspark.sql

Cmd 9

```python
#Creamos la tabla parsed_rdd.
#Vamos a importar la clase Row de la librería de pyspark.sql

from pyspark.sql import Row #Importamos la clase Row

parsed_rdd = csv_rdd.map(lambda r: Row(
    happiness_rank = r[0],
    country = r[1],
    happiness_score = r[2],
    gdp_per_capita = r[3],
    social_support = r[4],
    health = r[5],
    freedom = r[6],
    generosity = r[7],
    corruption = r[8]
    )
)
parsed_rdd.take(5)
```

(1) Spark Jobs

```
Out[23]: [Row(corruption='perceptions of corruption', country='country or region', freedom='freedom to make life choices', gdp_per_capita='gdp per capita', generosity='generosity', happiness_rank='overall rank', happiness_score='score', health='healthy life expectancy', social_support='social support'),
 Row(corruption='0.393', country='Finland', freedom='0.596', gdp_per_capita='1.34', generosity='0.153', happiness_rank='1', happiness_score='7.769', health='0.986', social_support='1.587'),
 Row(corruption='0.41', country='Denmark', freedom='0.592', gdp_per_capita='1.383', generosity='0.252', happiness_rank='2', happiness_score='7.6', health='0.996', social_support='1.573'),
 Row(corruption='0.341', country='Norway', freedom='0.603', gdp_per_capita='1.488', generosity='0.271', happiness_rank='3', happiness_score='7.554', health='1.028', social_support='1.582'),
 Row(corruption='0.118', country='Iceland', freedom='0.591', gdp_per_capita='1.38', generosity='0.354', happiness_rank='4', happiness_score='7.494', health='1.026', social_support='1.624')]
```

Command took 0.18 seconds -- by blancavazquez@comunidad.unam.mx at 2/5/2020, 9:04:22 AM on QuickHappiness

Monitoreo de actividades

```
1  print('No. de columnas: ', len(csv_rdd.take(1)[0]))
```

ched to cluster: QuickHappiness, 6.00 GB | 0.88
s | DBR 6.2 | Spark 2.4.4 | Scala 2.11

No. de columnas: 9

Command took 0.15 seconds -- by blancavazquez@comunidad.unam.mx at 2/5/2020, 8:45:27 AM on QuickHappiness

Cmd 9

```
1  #Creamos la tabla parsed_rdd.
2  #Vamos a importar la clase Row de la librería de pyspark.sql
3
4  from pyspark.sql import Row #Importamos la clase Row
5
6  parsed_rdd = csv_rdd.map(lambda r: Row(
7      happiness_rank = r[0],
8      country = r[1],
9      happiness_score = r[2],
10     gdp_per_capita = r[3],
11     social_support = r[4],
12     health = r[5],
13     freedom = r[6],
14     generosity = r[7],
15     corruption = r[8]
16     )
17 )
18 parsed_rdd.take(5)
```

(1) Spark Jobs
Job 36 View (Stages: 1/1)

```
Out[23]: [Row(corruption='perceptions of corruption', country='country or region', freedom='freedom t
s_score='score', health='healthy life expectancy', social_support='social support'),
 Row(corruption='0.393', country='Finland', freedom='0.596', gdp_per_capita='1.34', generosity='0.153
 Row(corruption='0.41', country='Denmark', freedom='0.592', gdp_per_capita='1.383', generosity='0.252
 Row(corruption='0.341', country='Norway', freedom='0.603', gdp_per_capita='1.488', generosity='0.271
 Row(corruption='0.118', country='Iceland', freedom='0.591', gdp_per_capita='1.38', generosity='0.354
```

Command took 0.18 seconds -- by blancavazquez@comunidad.unam.mx at 2/5/2020, 9:04:22 AM on QuickHappiness

Shift+Enter to run shortcuts

Jobs | Stages | Storage | Environment | Executors | SQL | JDBC/ODBC Server

Stages for All Jobs

Completed Stages: 37

Fair Scheduler Pools (1)

Pool Name	Minimum Share	Pool Weight	Active Stages	Running Tasks	SchedulingMode
default	0	1	0	0	FIFO

Completed Stages (37)

Stage Id ▼	Pool Name	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output
36	581849238594280514	#Creamos la tabla parsed_rdd. #Vamos a importar... runJob at PythonRDD.scala:186 +details	2020/02/05 15:04:22	96 ms	1/1	3.3 KB	
35	581849238594280514	#Creamos la tabla parsed_rdd. #Vamos a importar... runJob at PythonRDD.scala:186 +details	2020/02/05 15:03:18	0.1 s	1/1	3.3 KB	
34	581849238594280514	#Creamos la tabla parsed_rdd. #Vamos a importar... runJob at PythonRDD.scala:186 +details	2020/02/05 15:03:12	94 ms	1/1	3.3 KB	
33	581849238594280514	#Creamos la tabla parsed_rdd. #Vamos a importar... runJob at PythonRDD.scala:186 +details	2020/02/05 15:03:08	0.1 s	1/1	3.3 KB	
32	581849238594280514	#use specific columns from dataset from pyspar... runJob at PythonRDD.scala:186 +details	2020/02/05 14:58:55	94 ms	1/1	3.3 KB	
31	581849238594280514	#use specific columns from dataset from pyspar... runJob at PythonRDD.scala:186 +details	2020/02/05 14:58:39	0.1 s	1/1	3.3 KB	
30	581849238594280514	#En RDD, es necesario separar cada una de las e... runJob at PythonRDD.scala:186 +details	2020/02/05 14:57:09	0.1 s	1/1	3.3 KB	
29	581849238594280514	#use specific columns from dataset from pyspar... runJob at PythonRDD.scala:186 +details	2020/02/05 14:55:30	94 ms	1/1	3.3 KB	
28	581849238594280514	#use specific columns from dataset from pyspar... runJob at PythonRDD.scala:186 +details	2020/02/05 14:54:43	92 ms	1/1	3.3 KB	
27	581849238594280514	#use specific columns from dataset from pyspar... runJob at PythonRDD.scala:186 +details	2020/02/05 14:53:41	0.1 s	1/1	3.3 KB	
26	581849238594280514	print('No. de columnas: ', len(csv_rdd.take(1)[... runJob at PythonRDD.scala:186 +details	2020/02/05 14:45:27	97 ms	1/1	3.3 KB	
25	581849238594280514	print('potential # of columns: ', len(csv_rdd.t...	2020/02/05	0.1 s	1/1	3.3	

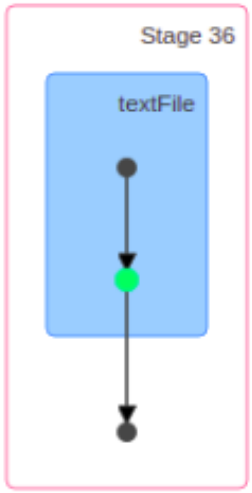
Monitoreo de actividades

Jobs Stages Storage Environment Executors SQL JDBC/ODBC Server

Details for Job 36

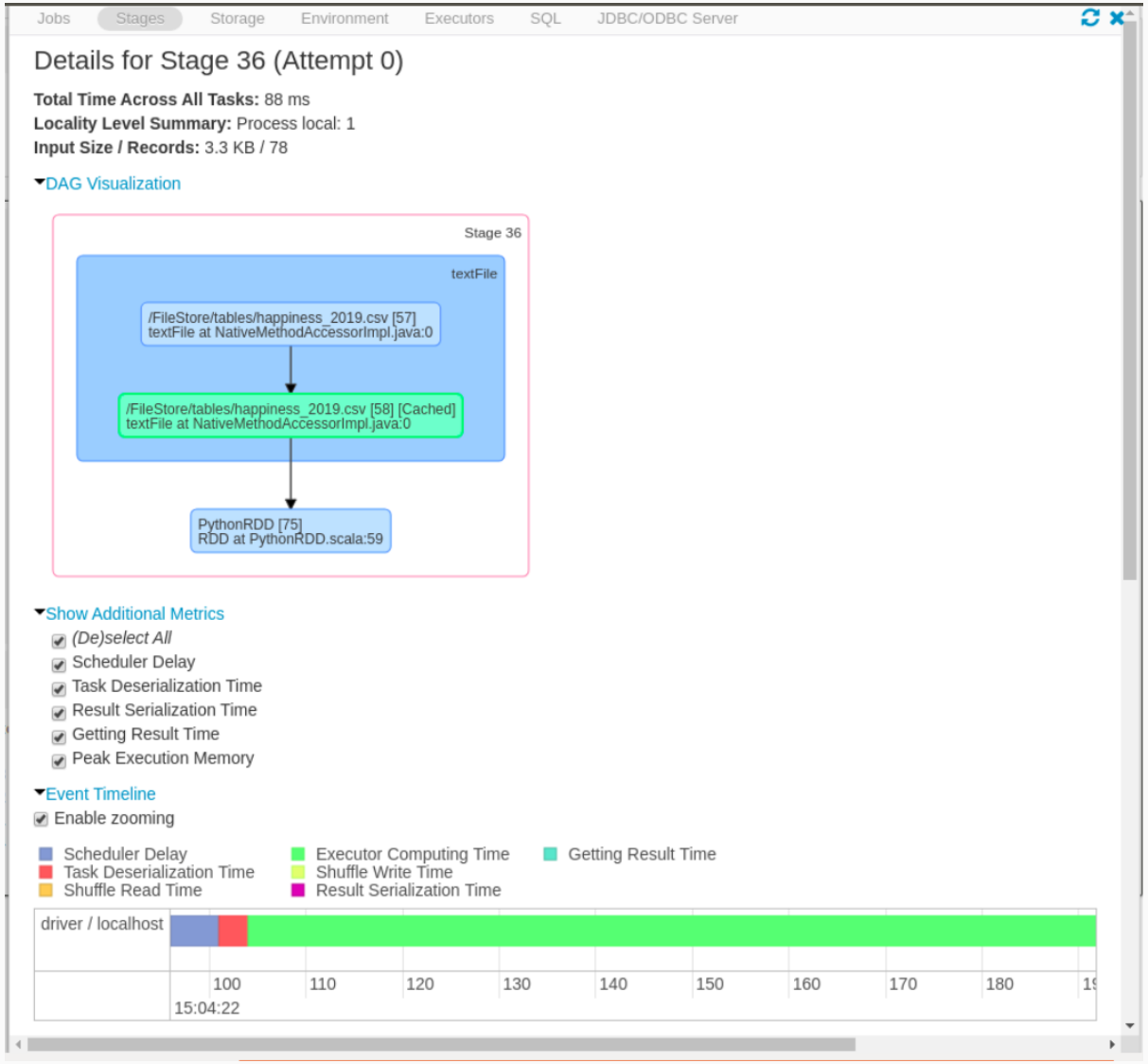
Status: SUCCEEDED
Job Group: 581849238594280514_4907341615727210820_9a5781fa58a14eca840116d870191d54
Completed Stages: 1

▸ Event Timeline
▾ DAG Visualization



▸ Completed Stages (1)

Monitoreo de actividades



Monitoreo de actividades

Jobs | Stages | Storage | Environment | Executors | SQL | JDBC/ODBC Server

PythonRDD [75]
RDD at PythonRDD.scala:59

Show Additional Metrics

- (De)select All
- Scheduler Delay
- Task Deserialization Time
- Result Serialization Time
- Getting Result Time
- Peak Execution Memory

Event Timeline

- Enable zooming

Scheduler Delay
Task Deserialization Time
Shuffle Read Time
Executor Computing Time
Shuffle Write Time
Result Serialization Time
Getting Result Time

driver / localhost

100 110 120 130 140 150 160 170 180
15:04:22

Summary Metrics for 1 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	88 ms	88 ms	88 ms	88 ms	88 ms
Scheduler Delay	5 ms	5 ms	5 ms	5 ms	5 ms
Task Deserialization Time	3 ms	3 ms	3 ms	3 ms	3 ms
GC Time	0 ms	0 ms	0 ms	0 ms	0 ms
Result Serialization Time	0 ms	0 ms	0 ms	0 ms	0 ms

Construcción de un dataframe

Cmd 10

```
#Creamos una tabla
temp_table_name = "happiness_2019_csv"
df.createOrReplaceTempView(temp_table_name)
```

Command took 0.04 seconds -- by blancavazquez@comunidad.unam.mx at 2/5/2020, 9:36:05 AM on QuickHappiness

Cmd 11

```
#Construimos un datraframe
df = sqlContext.createDataFrame(parsed_rdd)
display(df)
```

(3) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [corruption: string, country: string ... 7 more fields]

corruption	country	freedom	gdp_per_capita	generosity	happiness_rank	happiness_score	health	social_support
perceptions of corruption	country or region	freedom to make life choices	gdp per capita	generosity	overall rank	score	healthy life expectancy	social support
0.393	Finland	0.596	1.34	0.153	1	7.769	0.986	1.587
0.41	Denmark	0.592	1.383	0.252	2	7.6	0.996	1.573
0.341	Norway	0.603	1.488	0.271	3	7.554	1.028	1.582
0.118	Iceland	0.591	1.38	0.354	4	7.494	1.026	1.624
0.298	Netherlands	0.557	1.396	0.322	5	7.488	0.999	1.522
0.343	Switzerland	0.572	1.452	0.263	6	7.48	1.052	1.526
0.373	Sweden	0.574	1.387	0.267	7	7.343	1.009	1.487

Command took 0.52 seconds -- by blancavazquez@comunidad.unam.mx at 2/5/2020, 9:43:08 AM on QuickHappiness

1era consulta

Cmd 13

```
#Construimos una tabla temporal para correr comandos de SQL
#La tabla solo estará activa para esta sesión
df.registerTempTable("happiness")
```

Command took 0.05 seconds -- by blancavazquez@comunidad.unam.mx at 2/5/2020, 9:39:44 AM on QuickHappiness

Cmd 14

```
#Primera consulta: agrupamos los registros de la tabla por el campo de happiness_score (usan
display(df.groupBy('happiness_score')
          .count()
          .orderBy('count', ascending=False)
        )
```

(1) Spark Jobs

happiness_score	count
5.208	2
5.529	1
5.274	1
5.175	1
2.853	1
4.534	1
7.48	1
3.663	1
5.94	1

La misma consulta usando SQL

Cmd 15

```
#Misma consulta, ahora usando la sintáxis de SQL
happ_query = sqlContext.sql("""
                        SELECT happiness_score, count(*) as freq
                        FROM happiness
                        GROUP BY happiness_score
                        ORDER BY 2 DESC
                        """)
display(happ_query)
```

(1) Spark Jobs

happ_query: pyspark.sql.dataframe.DataFrame = [happiness_score: string, freq: long]

happiness_score	freq
5.208	2
5.529	1
5.274	1
5.175	1
2.853	1
4.534	1
6.321	1
7.48	1
2.662	1

Command took 1.41 seconds -- by blancavazquez@comunidad.unam.mx at 2/5/2020, 9:44:06 AM on QuickHappiness

¿Qué países tienen el mismo puntaje de felicidad?

Cmd 16

```
#Vamos a listar qué paises son los que tienen el mismo nivel de felicidad
happ_query = sqlContext.sql("""
                        SELECT country, happiness_score
                        FROM happiness
                        where happiness_score ==5.208
                        """)
display(happ_query)
```

(2) Spark Jobs

happ_query: pyspark.sql.dataframe.DataFrame = [country: string, happiness_score: string]

country	happiness_score
Morocco	5.208
Azerbaijan	5.208

Command took 0.27 seconds -- by blancavazquez@comunidad.unam.mx at 2/5/2020, 9:49:18 AM on QuickHappiness

2DA CONSULTA

Cmd 17

```
#Segunda consulta
happ_stats = sqlContext.sql("""
                            SELECT country, happiness_rank, corruption
                            FROM happiness
                            WHERE happiness_rank > 20
                            """)
display(happ_stats)
```

(2) Spark Jobs

happ_stats: pyspark.sql.dataframe.DataFrame = [country: string, happiness_rank: string ... 1 more fields]

country	happiness_rank	corruption
United Arab Emirates	21	0.182
Malta	22	0.151
Mexico	23	0.073
France	24	0.183
Taiwan	25	0.097
Chile	26	0.056
Guatemala	27	0.078
Saudi Arabia	28	0.132
Qatar	29	0.167

Command took 0.32 seconds -- by blancavazquez@comunidad.unam.mx at 2/5/2020, 9:50:56 AM on QuickHappiness

Guardar tabla

Cmd 18

```
#Guardamos la tabla de manera permanente para su posterior uso
permanent_table_name = "2019_csv"
df.write.format("parquet").saveAsTable(permanent_table_name)
```

▸ (1) Spark Jobs

Command took 4.85 seconds -- by blancavazquez@comunidad.unam.mx at 2/5/2020, 9:52:02 AM on QuickHappiness

Terminar clúster