

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 resilientes (RDD)**: es una colección de elementos particionados en los nodos del clúster para ser operados en paralelo.

DATOS DISTRIBUIDOS RESILIENTES (RDD)



sciabarra.com
JUST ADD DATA

Spark: Visualizing RDD transformations

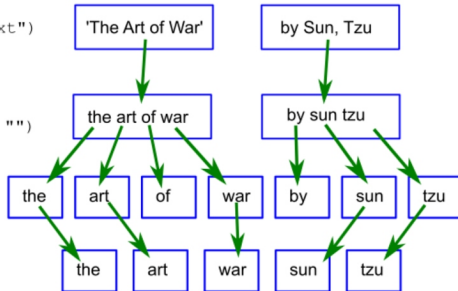
<http://sciabarra.com/blog/spark-visualizing-rdd>

```
sc.textFile("artofwar.txt")
```

```
.map(  
  _.toLowerCase  
  .replaceAll("[^\\w ]", "")  
)
```

```
.flatMap(_.split(" "))
```

```
.filter(_.size > 2)
```



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

The screenshot displays the Databricks website homepage. At the top, a navigation bar includes the Databricks logo and links for Platform, Solutions, Customers, Learn, Partners, Events, Open Source, and Company. On the right side of the navigation bar are links for English, Support, Contact, and Login, along with a 'TRY DATABRICKS' button. The main content area features the title 'Unified Data Analytics Platform' and the subtitle 'One cloud platform for massive scale data engineering and collaborative data science'. Below this, a central diagram illustrates the data engineering and science workflow. It shows a sequence of roles: DATA ENGINEERS, DATA SCIENTISTS, ML ENGINEERS, and DATA ANALYSTS, connected by arrows. The workflow is supported by various services and integrations, including miflow, DATA SCIENCE WORKSPACE, PYTORCH, BI INTEGRATIONS, UNIFIED DATA SERVICE, ENTERPRISE CLOUD SERVICE, and RAW DATA LAKE. The diagram also mentions integrations with AWS, Azure, and Delta Lake. The background of the diagram is a dark teal color with a subtle pattern of binary code.

<https://databricks.com/>

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

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

Home

Workspace

Recently

Data

Clusters

Jobs

Search

Upgrade ?

Welcome to databricks

Explore the Quickstart Tutorial

Spin up a cluster, run queries on preloaded data, and display results in 5 minutes.

Drop files or [click to browse](#)

Import & Explore Data

Quickly import data, preview its schema, create a table, and query it in a notebook.

Create a Blank Notebook

Create a notebook to start querying, visualizing, and modeling your data.

Common Tasks

- New Notebook
- Create Table
- New Cluster
- New Job
- New MLflow Experiment beta
- Import Library
- Read Documentation


Recents


- 2Happiness
- Happiness


What's new in v3.12


[View latest release notes](#)


CREACIÓN DE CLÚSTER



databricks



Home



Workspace


Recents


Data


Clusters


Jobs



Search

Create Cluster


New Cluster

Cancel

Create Cluster

0 Workers: 0.0 GB Memory, 0 Cores, 0 DBU
1 Driver: 6.0 GB Memory, 0.88 Cores, 1 DBU 

Cluster Name

Databricks Runtime Version 

Runtime: 6.2 (Scala 2.11, Spark 2.4.4) | v

New


This Runtime version supports only Python 3.

Instance

Free 6GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours. For [more configuration options](#), please [upgrade your Databricks subscription](#).

Instances

Spark

Availability Zone 

us-west-2c | v

databricks

Home

Workspace

Recents

Data

Clusters

Jobs

Create New Table (Python)

Data source ?

Upload File

S3

DBFS

Other Data Sources

Upload to DBFS ?

/FileStore/tables/ (optional)

Select

File ?

happiness_2019.csv
8.5 KB
[Remove file](#)

✓ File uploaded to /FileStore/tables/happiness_2019.csv

Create Table with UI

Create Table in Notebook ?

CARGAR DATOS

Specify Table Attributes

Specify the Table Name, Database and Schema to add this to the data UI for other users to access

Table Name ⓘ

happiness_2019_csv

Table name already exists in selected database

Create in Database ⓘ

default ⓘ

File Type ⓘ

CSV ⓘ

Columns Delimiter ⓘ

,

☒ First row is header ⓘ

☒ Infer schema ⓘ

☐ Multi-line ⓘ

Create Table

Table Preview

overall rank	country or region	score	gdp per capita	social support	healthy life expectancy	freedom to make life choice	generosity
INT	STRING	DOUBLE	DOUBLE	DOUBLE	DOUBLE	DOUBLE	DOUBLE
1	Finland	7.769	1.34	1.587	0.986	0.596	0.153
2	Denmark	7.6	1.383	1.573	0.996	0.592	0.252
3	Norway	7.554	1.488	1.582	1.028	0.603	0.271
4	Iceland	7.494	1.38	1.624	1.026	0.591	0.354
5	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322
6	Switzerland	7.48	1.452	1.526	1.052	0.572	0.263

The screenshot displays the Databricks Data interface. On the left is a dark sidebar with navigation icons and labels: databricks, Home, Workspace, Recents, Data, Clusters, Jobs, and Search. The main area is titled 'Data' and features two tabs: 'Databases' and 'Tables'. The 'Databases' tab is active, showing a search bar 'Filter Databases' and a list with 'default'. The 'Tables' tab is also visible, showing a search bar 'Filter Tables' and a list with 'diamonds' and 'happiness_2019_csv'. An 'Add Data' button and a pin icon are located in the top right corner of the main area.

Data

Databases

Filter Databases

default

Tables

Filter Tables

diamonds

happiness_2019_csv

Add Data

CREAR CUADERNO DE TRABAJO

The screenshot displays the Databricks Workspace interface. On the left is a dark sidebar with navigation icons and labels: databricks, Home, Workspace, Recents, Data, Clusters, Jobs, and Search. The main area is titled 'Workspace' and contains a list of items: Documentation, Release Notes, Training & Tutorials, Shared, Users, 1Diamonds, and 2Happiness. The '2Happiness' item is selected, and a context menu is open over it. The menu has a primary section with 'Create' (highlighted in blue), 'Import', 'Export', 'Permissions', and 'Sort'. The 'Create' submenu is expanded, showing options: 'Notebook' (highlighted in blue), 'Library', 'Folder', and 'MLflow Experiment'. Below the menu, a code editor shows a snippet of Scala code for reading a CSV file into a Spark DataFrame.

```
do Spark!  
(file_type) \  
, infer_schema) \  
st_row_is_header) \  
ter) \  
  
dataFrame  
ger  
string  
  
ble  
ble  
ancy: double
```

Create Notebook

Name

Language | v

Cluster | v

LECTURA DE DATOS USANDO APACHE SPARK

2Happiness (Python)

QuickHappiness

Cmd 1

```
1 #Opciones CSV
2 file_type = "csv"
3 infer_schema = "true"
4 first_row_is_header = "true"
5 delimiter = ","
6 file_location = "/FileStore/tables/happiness_2019.csv"
7
8 # Lectura de datos usando Spark!
9 df = spark.read.format(file_type) \
10     .option("inferSchema", infer_schema) \
11     .option("header", first_row_is_header) \
12     .option("sep", delimiter) \
13     .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))
```

Cell 3

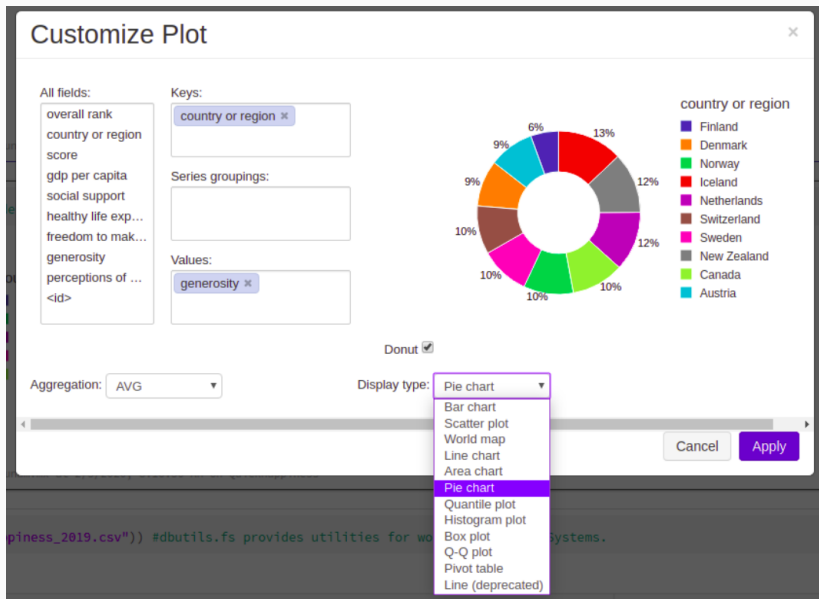
```
display(df.head(10)) #Muestra el contenido de los primeros 10 registros de la tabla happiness_2019.csv
```

4) Spark Job

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	1.395	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	1.397	1.497	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

Command took 1.65 seconds -- by blancavazquez@comunidad.unam.mx at 2/5/2020, 8:18:39 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 RESILIENTES (RDD)

Cnd 4

```
1 #Cargamos el archivo hacia un Resilient Distributed Dataset(RDD)
2 data_file = "/FileStore/tables/happiness_2019.csv"
3 raw_rdd = sc.textFile(data_file).cache()
4 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

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

```
1 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

```
1 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

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: <code>pyspark</code> :: Module: <code>rdd</code> :: Class: <code>RDD</code>		Classpath / maven
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		
	<code>__init__(self, jobs, cte)</code> X__init__(...) initializes x; see <code>help(type(x))</code> for signature	Source code
	<code>cache(self)</code> Persist this RDD with the default storage level (<code>MEMORY_ONLY</code>).	Source code
	<code>persist(self, storageLevel)</code> Set this RDD's storage level to persist its values across operations after the first time it is computed.	Source code
	<code>unpersist(self)</code> Mark the RDD as non-persistent, and remove all blocks for it from memory and disk.	Source code
	<code>checkpoint(self)</code> Mark this RDD for checkpointing.	Source code
	<code>isCheckpointed(self)</code> Return whether this RDD has been checkpointed or not	Source code
	<code>getCheckpointFile(self)</code> Gets the name of the file to which this RDD was checkpointed	Source code
	<code>map(self, f, preservesPartitioning=False)</code> 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 PYSARK.SQL

Cmd 9

```
1 #Creamos la tabla parsed_rdd.
2 #Vamos a importar la clase Row de la libreria 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

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='0.683', 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

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

Need to cluster: QuickHappiness, 6.00 GB | 0.88
6 | OMR 5.2 | Spark 2.4.4 | Scala 2.11
New user: custer@ms: ~
Command took 8-15 seconds -- by blancaaquino@comfide.uan.mx on 2/5/2020, 8:45:27 AM on QuickHappiness

End 0

1 #Creamos la tabla parsed_rdd.
2 #Vamos a importar la clase Row de la libreria 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) Spain Job
# Job 36 View [Stages: 17]

Out[13]: Row(corruption='perceptions of corruption', country='country or region', freedom='freedom to
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='0.42', country='Denmark', freedom='0.592', gdp_per_capita='1.383', generosity='0.252
Row(corruption='0.341', country='Norway', freedom='0.683', 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 8-18 seconds -- by blancaaquino@comfide.uan.mx on 2/5/2020, 9:04:22 AM on QuickHappiness

Shift+Enter to run shortcuts
```

Jobs	Stages	Storage	Environment	Executors	SQL	JOB-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					
30	561849238594280514	Acremos la tabla parsed_rdd. #Vamos a importar... runJob at PythonRDD scala:186	2020/02/05 15:04:22	90 ms	1/1	3.3	KB							
35	561849238594280514	Acremos la tabla parsed_rdd. #Vamos a importar... runJob at PythonRDD scala:186	2020/02/05 15:03:18	0.1 s	1/1	3.3	KB							
34	561849238594280514	Acremos la tabla parsed_rdd. #Vamos a importar... runJob at PythonRDD scala:186	2020/02/05 15:03:12	94 ms	1/1	3.3	KB							
33	561849238594280514	Acremos la tabla parsed_rdd. #Vamos a importar... runJob at PythonRDD scala:186	2020/02/05 15:03:08	0.1 s	1/1	3.3	KB							
32	561849238594280514	Ause specific columns from dataset from pyspar... runJob at PythonRDD scala:186	2020/02/05 14:58:55	94 ms	1/1	3.3	KB							
31	561849238594280514	Ause specific columns from dataset from pyspar... runJob at PythonRDD scala:186	2020/02/05 14:58:39	0.1 s	1/1	3.3	KB							
30	561849238594280514	#En RDD, es necesario separar cada una de las e... runJob at PythonRDD scala:186	2020/02/05 14:57:09	0.1 s	1/1	3.3	KB							
29	561849238594280514	Ause specific columns from dataset from pyspar... runJob at PythonRDD scala:186	2020/02/05 14:55:30	94 ms	1/1	3.3	KB							
28	561849238594280514	Ause specific columns from dataset from pyspar... runJob at PythonRDD scala:186	2020/02/05 14:54:43	92 ms	1/1	3.3	KB							
27	561849238594280514	Ause specific columns from dataset from pyspar... runJob at PythonRDD scala:186	2020/02/05 14:53:41	0.1 s	1/1	3.3	KB							
26	561849238594280514	print("No. de columnas : ", len(scs_rdd.take(1))... runJob at PythonRDD scala:186	2020/02/05 14:45:27	97 ms	1/1	3.3	KB							
25	561849238594280514	print("potential # of columns : ", len(scs_rdd.i...	2020/02/05 0.1 s		1/1	3.3	KB							

[Jobs](#) [Stages](#) [Storage](#) [Environment](#) [Executors](#) [SQL](#) [JDBC/ODBC Server](#)

Details for Job 36

Status: SUCCEEDED

Job Group: 581849238594280514_4907341615727210820_9a5781fa58a14eca840116d870191d54

Completed Stages: 1

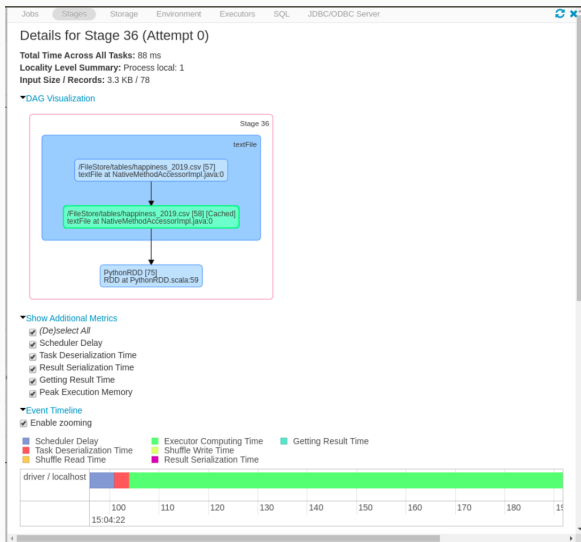
- ▶ [Event Timeline](#)
- ▼ [DAG Visualization](#)

Stage 36

The diagram shows a single task node labeled 'textFile' inside a light blue rounded rectangle. A vertical line with a downward arrow connects a black dot at the top to a green dot in the middle, and then to another black dot at the bottom. The green dot indicates that the task is completed. The entire task node is enclosed in a pink rectangular border.

- ▶ [Completed Stages \(1\)](#)

MONITOREO DE ACTIVIDADES



MONITOREO DE ACTIVIDADES



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

```
1 #Creamos una tabla
2 temp_table_name = "happiness_2019_csv"
3 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

```
1 #Construimos un dataframe
2 df = sqlContext.createDataFrame(parsed_rdd)
3 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

```
1 #Construimos una tabla temporal para correr comandos de SQL
2 #La tabla solo estará activa para esta sesión
3 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

```
1 #Primera consulta: agrupamos los registros de la tabla por el campo de happiness_score (usan
2 display(df.groupBy('happiness_score')
3           .count()
4           .orderBy('count', ascending=False)
5           )
```

► (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
5.04	1

LA MISMA CONSULTA USANDO SQL

Cmd 15

```
1 #Misma consulta, ahora usando la sintaxis de SQL
2 happ_query = sqlContext.sql("""
3     SELECT happiness_score, count(*) as freq
4     FROM happiness
5     GROUP BY happiness_score
6     ORDER BY 2 DESC
7     """)
8 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

```
1 #Vamos a listar qué países son los que tienen el mismo nivel de felicidad
2 happ_query = sqlContext.sql("""
3     SELECT country, happiness_score
4     FROM happiness
5     where happiness_score ==5.208
6     """)
7 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

```
1 #Segunda consulta
2 happ_stats = sqlContext.sql("""
3     SELECT country, happiness_rank, corruption
4     FROM happiness
5     WHERE happiness_rank > 20
6     """)
7 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
Oman	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

```
1 #Guardamos la tabla de manera permanente para su posterior uso
2 permanent_table_name = "2019_csv"
3 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

The screenshot shows the Databricks web interface for configuring a new cluster. On the left is a dark sidebar with navigation icons and labels: Databricks, Home, Workspace, Recents, Data, Clusters, Jobs, and Search. The main content area is titled 'Clusters / Quick'. Below the title is a 'Quick' section with a green circle icon and five buttons: 'Edit', 'Clone', 'Restart', 'Terminate' (highlighted with a dark background), and 'Delete'. A horizontal menu below these buttons includes 'Configuration' (underlined), 'Notebooks (1)', 'Libraries', 'Event Log', 'Spark UI', 'Driver Logs', 'Metrics', 'Apps', and 'Spark Cluster UI - Master'. The 'Configuration' section contains three fields: 'Databricks Runtime Version' set to '6.2 (includes Apache Spark 2.4.4, Scala 2.11)', 'Driver Type' set to 'Community Optimized' with a subtext '6.0 GB Memory, 0.88 Cores, 1 DBU', and 'Instance' with a warning box stating 'Free 6GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours. For more configuration options, please upgrade your Databricks subscription.' Below these fields is a sub-menu with 'Instances' (underlined), 'Spark', 'JDBC/ODBC', and 'Permissions'. The 'Availability Zone' field is set to 'us-west-2c'.

Clusters / Quick

Quick Edit Clone Restart Terminate Delete

Configuration Notebooks (1) Libraries Event Log Spark UI Driver Logs Metrics Apps Spark Cluster UI - Master ▼

Databricks Runtime Version

6.2 (includes Apache Spark 2.4.4, Scala 2.11)

New This Runtime version supports only Python 3.

Driver Type

Community Optimized 6.0 GB Memory, 0.88 Cores, 1 DBU

Instance

Free 6GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours. For more configuration options, please upgrade your Databricks subscription.

Instances Spark JDBC/ODBC Permissions

Availability Zone ⓘ

us-west-2c