Notas Curso:  
**Big Data y Spark: ingeniería de datos con Python y pyspark**

**Jose Maximiliano Retana Lopez**

Table of Contents

[Introducción 4](#_Toc157160949)

[Google Colab (Mi cuenta) 5](#_Toc157160950)

[Descargar e instalar Spark en Colab 6](#_Toc157160951)

[SparkSession 11](#_Toc157160952)

[¿Que es un RDD? 13](#_Toc157160953)

[Transformaciones en un RDD 14](#_Toc157160954)

[Introducción 14](#_Toc157160955)

[Tipos de Transformaciones 15](#_Toc157160956)

[Transformaciones: función map 16](#_Toc157160957)

[Codigo Función Map 17](#_Toc157160958)

[Transformaciones: Función flatMap 17](#_Toc157160959)

[Codigo Función flatMap 17](#_Toc157160960)

[Transformaciones: Función filter 17](#_Toc157160961)

[Codigo Función filter 18](#_Toc157160962)

[Transformaciones: Función coalesce 18](#_Toc157160963)

[Código Función coalesce 18](#_Toc157160964)

[Transformaciones: Función repartition 18](#_Toc157160965)

[Codigo Función repartition 18](#_Toc157160966)

[Transformacion: Función reduceByKey 19](#_Toc157160967)

[Codigo Función reduceByKey 19](#_Toc157160968)

[Acciones sobre un RDD en Spark 19](#_Toc157160969)

[Introducción 19](#_Toc157160970)

[Tipos de acciones 19](#_Toc157160971)

[Driver 19](#_Toc157160972)

[Distribuidas 20](#_Toc157160973)

[Función reduce 20](#_Toc157160974)

[Codigo Función reduce 20](#_Toc157160975)

[Función count 20](#_Toc157160976)

[Codigo Función count 21](#_Toc157160977)

[Función collect 21](#_Toc157160978)

[Codigo Función collect 21](#_Toc157160979)

[Funciones take, max y saveAsTextFile 21](#_Toc157160980)

[Codigo Funciones take, max y saveAsTextFile 21](#_Toc157160981)

[Aspectos avanzados sobre RDD 21](#_Toc157160982)

[Almacenamiento en cache 21](#_Toc157160983)

[Codigo 22](#_Toc157160984)

[Particionando 23](#_Toc157160985)

[Numero de particiones 23](#_Toc157160986)

[Particionadores 24](#_Toc157160987)

[Codigo 25](#_Toc157160988)

[Mezcla de datos(shuffling) 26](#_Toc157160989)

[Ejemplo 26](#_Toc157160990)

[Broadcast variables 26](#_Toc157160991)

[Codigo 26](#_Toc157160992)

[Acomuladores 27](#_Toc157160993)

[Codigo 27](#_Toc157160994)

# Introducción

Científicos de datos:

Analizan y modelan los datos que se descubren

Tienen experiencia en SQL, estadística, modelado, predictivo y programación

Su objetivo es responder una pregunta

Ingeniero de datos

Comprenden los principios de la ingeniería de software

# Google Colab (Mi cuenta)

Primer paso, crear una cuanta de Gmail, en el caso de no contar con una.

Correo: mretanaHyperox@gmail.com

Contrasena: r3t4n4Hyp3r0x

# Descargar e instalar Spark en Colab

Paso 1

# Instalar SDK Java 8

> !apt-get install openjdk-8-jdk-headless -qq > /dev/null

Paso 2

# Descargar Spark (*La versión que nos sea útil*)

> !wget -q <https://archive.apache.org/dist/spark/spark-3.5.0/spark-3.5.0-bin-hadoop3.tgz>

Paso 3

# Descomprimir la version de Spark

> !tar xf spark-3.5.0-bin-hadoop3.tgz

Paso 4

# Establecer las variables de entorno

> import os

> os.environ["JAVA\_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"

> os.environ["SPARK\_HOME"] = "/content/spark-3.5.0-bin-hadoop3"

Paso 5

# Instalar la librería findspark

> !pip install -q findspark

Paso 6

# Instalar pyspark

> !pip install -q pyspark

Paso 7

### Verificar la instalación ###

> import findspark

> findspark.init()

> from pyspark.sql import SparkSession

> spark = SparkSession.builder.master("local[\*]").getOrCreate()

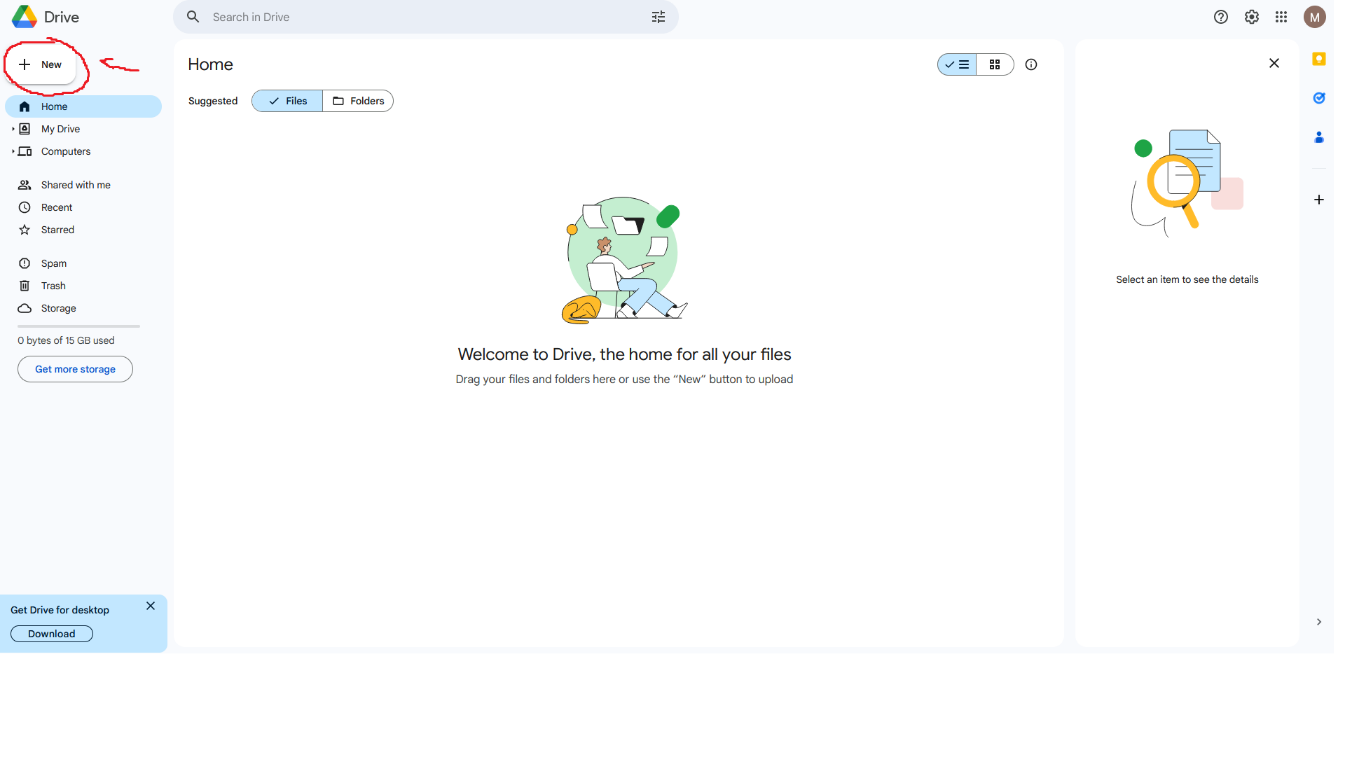
Paso 8

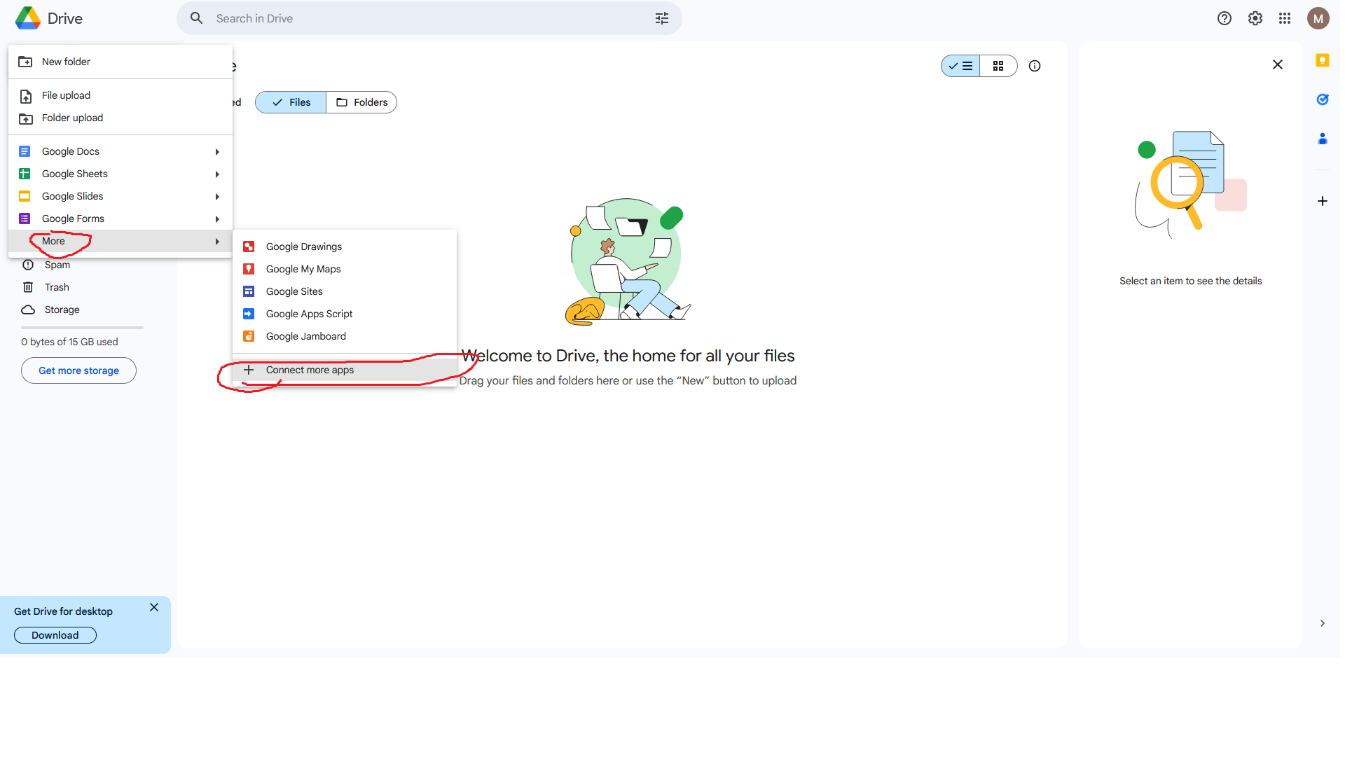
# Probando la sesión de Spark

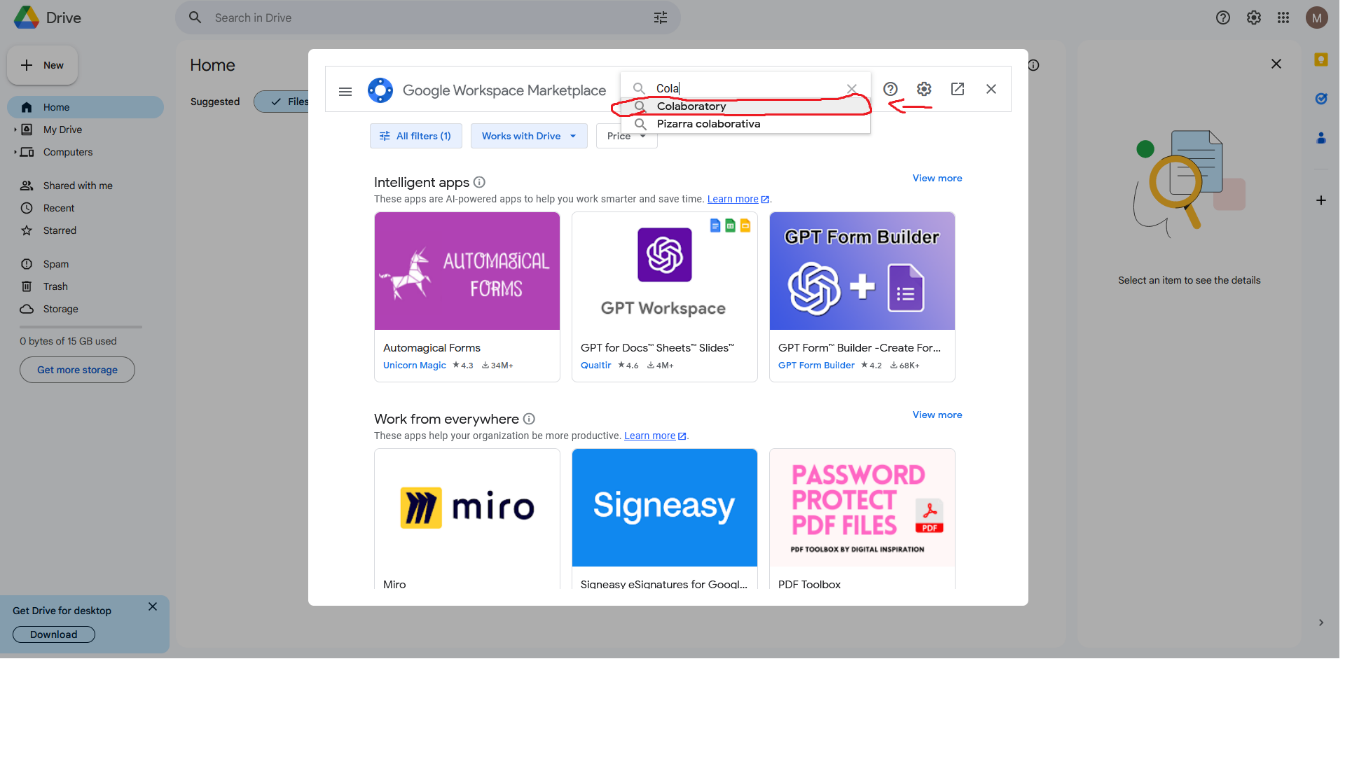
> df = spark.createDataFrame([{"Hola": "Mundo"} for x in range(10)])

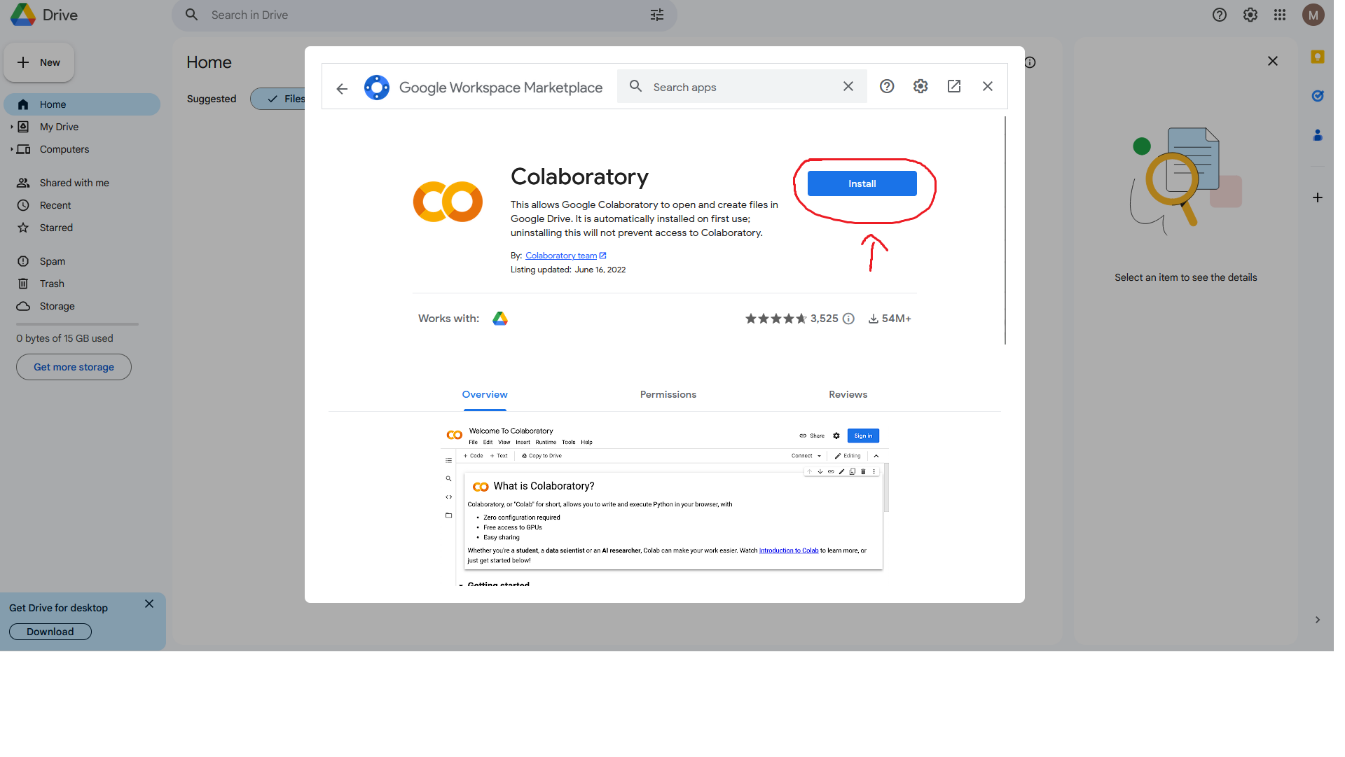
> df.show(10, False)

Después instalamos Google colab









Componentes a considerar

Indice o trabla de contendios

A screen shot of a computer

Description automatically generated

Nos muestra los diferentes títulos que vayamos creando a lo largo del documento

Buscador

A screenshot of a computer

Description automatically generated

Nos ayuda a buscar o remplazar alguna palabra que tengamos en nuestro documento

archivos

A close up of a computer screen

Description automatically generated

Aquí subiremos archivos de apoyo para cuando realicemos códigos que lo necesiten

# SparkSession

A diagram of a diagram

Description automatically generated

Crear sesión

> import findspark

> findspark.init()

> from pyspark.sql import SparkSession

> spark = SparkSession.builder.getOrCreate()

Ejemplo

A screenshot of a computer program

Description automatically generated

Para indicarle a la session de spark con que recursos va a correr la aplicación de spark

Ejecutandolo de manera local

> import findspark

> findspark.init()

> from pyspark.sql import SparkSession

> spark = SparkSession.builder.master(“local[\*]”).appName(‘Curso Pyspark’).getOrCreate()

# ¿Que es un RDD?

Resilient Distributed Dataset (RDD) es la abstracción principal de Spark

Caractersticas Asociadas

Dependencias

Particiones

Función de calculo

# Transformaciones en un RDD

## Introducción

Los RDD son inmutables y cada operación crea un nuevo RDD

Principales operaciones que se pueden realizar sobre un RDD

* Transformaciones
* Acciones

**Transformaciones**

* Dividir el elemento de entrada
* Filtrar elementos
* Realizar cálculos de algún tipo

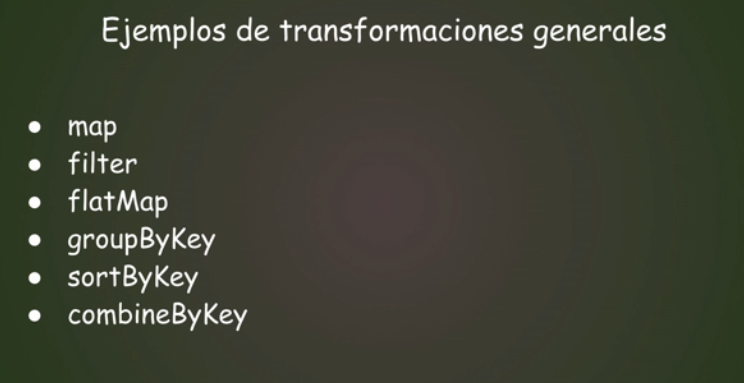


## Tipos de Transformaciones

Categorías de las transformaciones

* Transformaciones generales
  + son funciones de transformación que manejan la mayoría de los casos de uso de propósito general. Estas aplican la lógica de transformaciones a los RDD existentes y generan un nuevo RDD. Las operaciones comunes de agregación, filtros, etc. Se conocen como transformaciones generales.
* Transformaciones matemáticas o estadísticas
  + Son funciones de transformación que manejan alguna funcionalidad estadística y que generalmente aplican alguna operación matemática o estadística en RDD existente, generando un nuevo RDD. El muestreo es un gran ejemplo de esto y se usa a menudo en las aplicaciones de Spark.
* Transformaciones de conjunto o relacionales
  + Son funciones de transformación que manejan transformaciones como uniones de conjuntos de datos y otras funciones algebraicas relacionales como grupo. Esta funciona aplicando la lógica de transformación a los RDD existentes y generando un nuevo RDD.
* Transformaciones basadas en estructura de datos
  + Son funciones de transformación que operan en las estructuras de datos subyacentes del RDD. Estas funciones pueden trabajar directamente en particiones sin tocar directamente los elementos o los datos dentro del RDD.

Ejemplos



A black background with white text

Description automatically generatedA black board with white text

Description automatically generatedA black background with white text

Description automatically generated

## Transformaciones: función map

Map aplica la función de transformación que le proporcionamos a las particiones de entrada para generar particiones de salida en el RD de salida.

A diagram of a map

Description automatically generated

### Codigo Función Map

<https://colab.research.google.com/drive/1qhEJ4CxttjnJmK6H7Tj7qI0gD9h1UE_m?usp=drive_link>

## Transformaciones: Función flatMap

Aplica la función de transformación a las particiones de entrada para generar particiones de salida en el RDD de salida.

FlatMap también aplana o hace un floating de cualquier colección en los elementos del RD de entrada.

A diagram of a diagram

Description automatically generated

### Codigo Función flatMap

<https://colab.research.google.com/drive/1WS9AnyohdeUAre89MsdcNiu3bN7G7qCU?usp=sharing>

## Transformaciones: Función filter

Filter aplica la función de transformación a las particiones de entrada para generar particiones de salida filtradas en el RDD de salida.

**A diagram of a diagram

Description automatically generated with medium confidence**

### Codigo Función filter

<https://colab.research.google.com/drive/1qTRVLgvRV7tBhtNMwMwn7myECRBazK2I?usp=sharing>

## Transformaciones: Función coalesce

Aplica una función de transformación a las particiones de entrada para combinar las particiones de entrada en menos particiones en el RDD de salida.

A diagram of a diagram

Description automatically generated

### Código Función coalesce

<https://colab.research.google.com/drive/1_QQhxvj_ABHa7kR_T6M46dw876q8PDS3?usp=sharing>

## Transformaciones: Función repartition

Aplica una función de transformación a las particiones de entrada para repartir la entrada en menos o más particiones de salida en el RDD de salida.

A diagram of a process

Description automatically generated

### Codigo Función repartition

[https://colab.research.google.com/drive/1zzeDvcwv8Pm q38CkDUng9Ewa8GBy2Xa?usp=sharing](https://colab.research.google.com/drive/1zzeDvcwv8Pm%20q38CkDUng9Ewa8GBy2Xa?usp=sharing)

## Transformacion: Función reduceByKey

Se usa para fusionar los valores de cada clave usando una función asociativa de reducción. Es una transformación de tipo amplia o builder, ya que intercambia datos en múltiples particiones y opera por el par llave valor.

A diagram of a program

Description automatically generated with medium confidence

### Codigo Función reduceByKey

<https://colab.research.google.com/drive/1_Dq8w23daVINje7zeEa4zvhWR9HcV6bJ?usp=sharing>

# Acciones sobre un RDD en Spark

## Introducción

Las acciones son operaciones que en realidad activan los cálculos hasta que se encuentra una operación de acción. El plan de ejecución dentro del programa Spark se crea en forma de DAC.

## Tipos de acciones

### Driver

Estas suelen recopilar conteos, recuentos por tecla, etc. Cada una de estas acciones realiza algunos cálculos en el ejecutor remoto y devuelve los datos al controlador.

### Distribuidas

Son los que se ejecutan en los nodos del cluster. Un ejemplo de tal acción distribuida es Safe Attacks Failed. Esta es la operación de acción más común debido a la naturaleza distribuida deseable de la misma operación.

## Función reduce

Reduce aplica la función de reducción a todos los elementos del RD y la envía al driver.

A diagram of a computer program

Description automatically generated

### Codigo Función reduce

<https://colab.research.google.com/drive/1fGrWJJ1pvyLqrB_a_AsIT36mkjl0dyuN?usp=sharing>

## Función count

Simplemente cuenta el número de elementos en el RD y lo envía al controlador.



### Codigo Función count

<https://colab.research.google.com/drive/1OjGql58QE3KvqBb04V8oZtzPxDCFULHP?usp=sharing>

## Función collect

Recopila todos los elementos del RD y lo envía al controlador.

A computer screen shot of a computer

Description automatically generated

### Codigo Función collect

<https://colab.research.google.com/drive/1f1t2gX1p8IAbYJq0WSRyhNRodfl0i8PT?usp=sharing>

## Funciones take, max y saveAsTextFile

### Codigo Funciones take, max y saveAsTextFile

<https://colab.research.google.com/drive/1xiTXzL_5jG8GULyqmZ2c15DqAW-jFctg?usp=sharing>

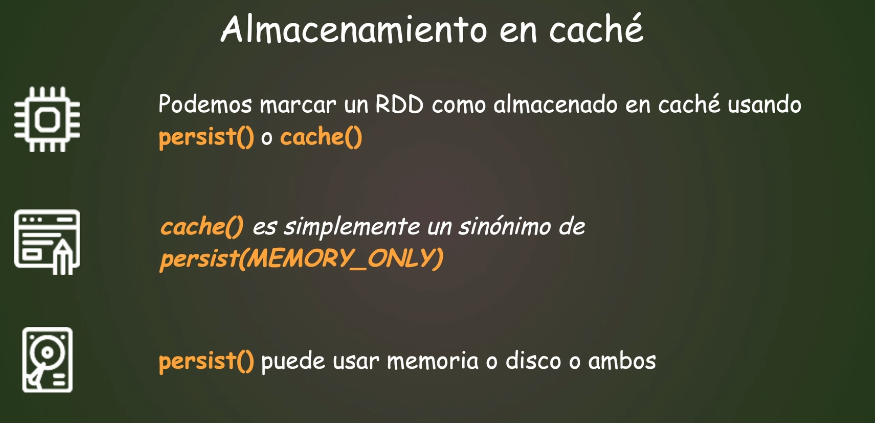
# Aspectos avanzados sobre RDD

## Almacenamiento en cache

El almacenamiento en caché permite que Spark conserve los datos en todos los cálculos y operaciones. De hecho, esta es una de las técnicas más importantes de Spark para acelerar los cálculos, especialmente cuando se trata de cálculos interactivos.

El almacenamiento en caché funciona almacenando el RDD tanto como sea posible en la memoria.

Si los datos que se solicitan para almacenar en caché son más grandes que la memoria disponible, el rendimiento disminuirá porque se utilizará disco en lugar de memoria.

  
A black board with white text

Description automatically generated

A black and white text on a black background

Description automatically generated

### Codigo

<https://colab.research.google.com/drive/1bchWhSP3Ss5k1Cx6mreNsVA595I175kg?usp=sharing>

## Particionando

Los RDD operan con datos no como una sola masa de datos, sino que administran y operan los datos en particiones repartidas por todo el cluster.

Por lo tanto, el concepto de partición de datos es fundamental para el correcto funcionamiento de los jobs de Apache Spark y puede tener un gran efecto en el rendimiento y en la forma en que se utilizan los recursos

A black screen with white text

Description automatically generated

### Numero de particiones

A black and white text on a black background

Description automatically generated

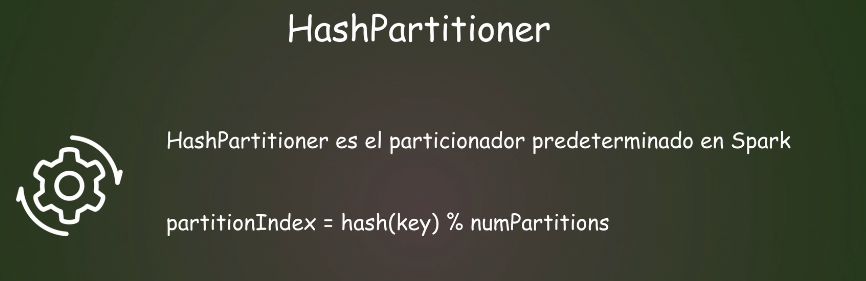
### Particionadores

A diagram of a graph

Description automatically generated

#### HashPartitioner

Es el patrocinador predeterminado en Spark y funciona calculando un valor para cada clave de los elementos. Todos los elementos con el mismo código terminan en una misma partición y esto se hace utilizando la fórmula siguiente.



##### Como funciona

A diagram of a graph

Description automatically generated

#### RangePartitioner

Funciona dividiendo el RDD en rangos aproximadamente iguales, dado que el rango debe conocer las claves de inicio y final de cualquier partición. El RDD debe ordenarse primero antes de que se pueda usar un rango particular.

A black and white text on a black background

Description automatically generated

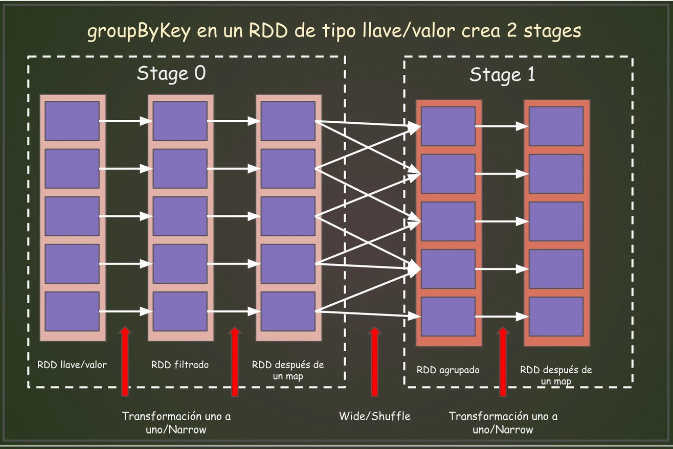
### Codigo

<https://colab.research.google.com/drive/1AOMqXNSOy--62jLaoZTTVJpJMTjfTLCo?usp=sharing>

## Mezcla de datos(shuffling)

El movimiento de datos necesario para el reparticionamiento se denomina shuffling

### Ejemplo



## Broadcast variables

Las variables broadcast son variables compartidas entre todos los ejecutores. Éstas se crean una vez en el controlador y luego se leen solo los ejecutores. Se pueden transmitir conjuntos de datos completos en un cluster de Spark para que los ejecutores tengan acceso a los datos transmitidos. Todas las tareas que se ejecutan dentro de un ejecutor tienen acceso a las variables broadcast.

A screenshot of a computer

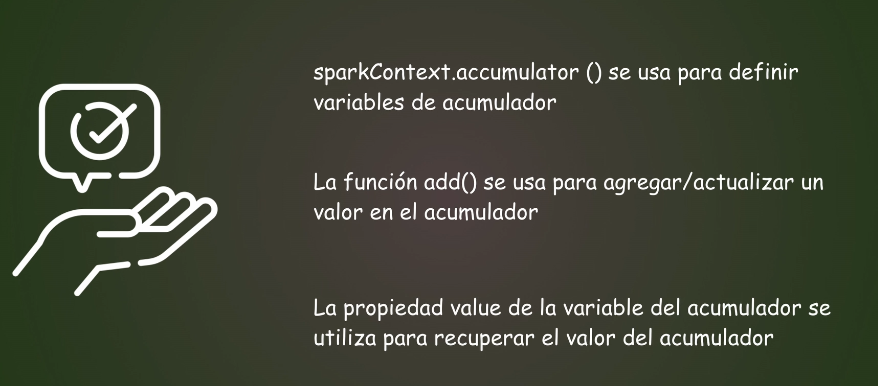
Description automatically generated

### Codigo

<https://colab.research.google.com/drive/1sKfjcaGgHumewQrj-6hDV37UyDhc4AYv?usp=sharing>

## Acomuladores

Los acumuladores son variables compartidas entre los ejecutores que normalmente se usan para agregar contadores a su programa en Spark.

Puntos a tener en cuenta  


### Codigo

<https://colab.research.google.com/drive/1I8zgQ2BvSVtoYOf4AZLSIYDQbB1ufmWd?usp=sharing>