# EJERCICIOS

## EJERCICIOS PARA SPARK Y HADOOP EN UBUNTU (USANDO SPARK-SUBMIT)

### EJERCICIO 1

Calcula el número promedio de palabras por línea en un fichero de texto que tengas en Hadoop y guarda el resultado en un fichero (en el sistema operativo, no en Hadoop).

from pyspark.sql import SparkSession

# Crear sesión de Spark

spark = SparkSession.builder.appName("PromedioPalabrasPorLinea").getOrCreate()

# Cargar el archivo desde HDFS

archivo\_hdfs = "hdfs://master:9000/quijote.txt"

rdd = spark.sparkContext.textFile(archivo\_hdfs)

# Contar el número de líneas

num\_lineas = rdd.count()

# Contar el total de palabras en el archivo

num\_palabras = rdd.flatMap(lambda linea: linea.split()).count()

# Calcular el promedio de palabras por línea

if num\_lineas > 0:

promedio = num\_palabras / num\_lineas

else:

promedio = 0

# Guardar el resultado en un archivo local

with open("resultado.txt", "w") as f:

f.write(f"Promedio de palabras por línea: {promedio:.2f}\n")

# Mostrar resultado en consola

print(f"Promedio de palabras por línea: {promedio:.2f}")

# Cerrar sesión de Spark

spark.stop()





### EJERCICIO 2

Calcula la suma acumulada de números en un archivo de texto (que tiene que estar en local, no en hadoop), donde cada línea contiene un número (tendrás que crear el archivo con los números también). Guarda el resultado en un fichero (en el sistema operativo, no en Hadoop)

from pyspark.sql import SparkSession

from pyspark.rdd import RDD

# Crear la sesión de Spark

spark = SparkSession.builder.appName("SumaAcumulada").getOrCreate()

# Leer el archivo de texto desde el sistema local

archivo = "numeros.txt"

rdd = spark.sparkContext.textFile(archivo)

# Convertir las líneas en números

numeros\_rdd = rdd.map(lambda x: int(x))

# Calcular la suma acumulada

suma\_acumulada\_rdd = numeros\_rdd.mapPartitions(lambda it: [sum(it)])

# Convertir el resultado a una lista

resultado = suma\_acumulada\_rdd.collect()

# Guardar el resultado en un archivo local

with open("resultado\_suma\_acumulada.txt", "w") as f:

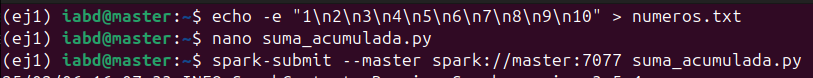
for valor in resultado:

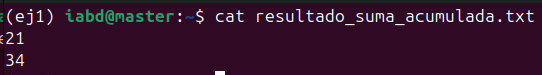
f.write(str(valor) + "\n")

print("Resultado guardado en 'resultado\_suma\_acumulada.txt'")

# Detener Spark

spark.stop()





## EJERCICIOS PARA SPARK CON DOCKER EN UBUNTU (USANDO SPARK-SUBMIT)

### EJERCICIO 3

Realiza un join entre dos RDDs donde el primero contiene el id y el nombre de varias personas (7 por lo menos) y el segundo el id y la edad y selecciona las entradas donde la edad es mayor a 30.

from pyspark import SparkContext

# Inicializar el SparkContext

sc = SparkContext(appName="JoinExample")

# Crear dos RDDs

rdd1 = sc.parallelize([(1, "Alice"), (2, "Bob"), (3, "Charlie"), (4, "David"), (5, "Eve"), (6, "Frank"), (7, "Grace")])

rdd2 = sc.parallelize([(1, 25), (2, 30), (3, 35), (4, 40), (5, 45), (6, 50), (7, 55)])

# Realizar el join entre los dos RDDs

joined\_rdd = rdd1.join(rdd2)

# Filtrar solo los resultados donde la edad sea mayor a 30

filtered\_rdd = joined\_rdd.filter(lambda x: x[1][1] > 30)

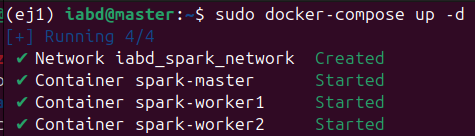
# Mostrar el resultado

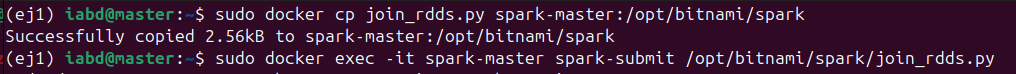
for record in filtered\_rdd.collect():

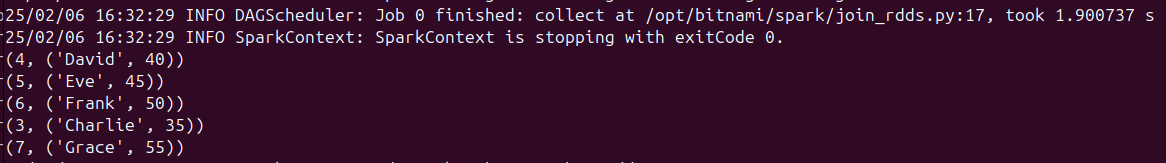
print(record)

# Detener el SparkContext

sc.stop()







### EJERCICIO 4

Dado un RDD con tuplas (categoria, valor), calcula el valor medio por categoría (tendrás que crear un rdd con al menos 11 líneas).

from pyspark import SparkContext

# Inicializa el contexto de Spark

sc = SparkContext("local", "Mean Per Category")

# Crea un RDD con tuplas (categoria, valor)

rdd = sc.parallelize([

("A", 10),

("B", 20),

("A", 30),

("C", 40),

("B", 30),

("C", 50),

("A", 50),

("C", 60),

("B", 40),

("A", 20),

("B", 10)

])

# Paso 1: Crea un RDD con el formato (categoria, (valor, 1)) para sumar y contar

rdd1 = rdd.map(lambda x: (x[0], (float(x[1]), 1)))

# Paso 2: Utiliza reduceByKey para sumar los valores y contar las ocurrencias por categoría

rdd2 = rdd1.reduceByKey(lambda a, b: (a[0] + b[0], a[1] + b[1]))

# Paso 3: Calcula el promedio dividiendo la suma entre el contador

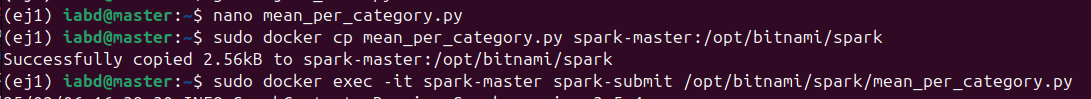
rdd3 = rdd2.mapValues(lambda x: x[0] / x[1])

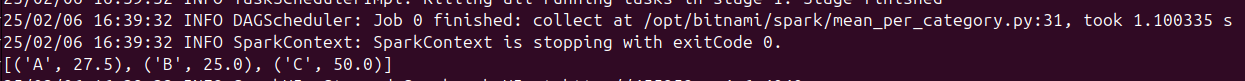
# Muestra el resultado

print(rdd3.collect())

# Detén el contexto de Spark

sc.stop()



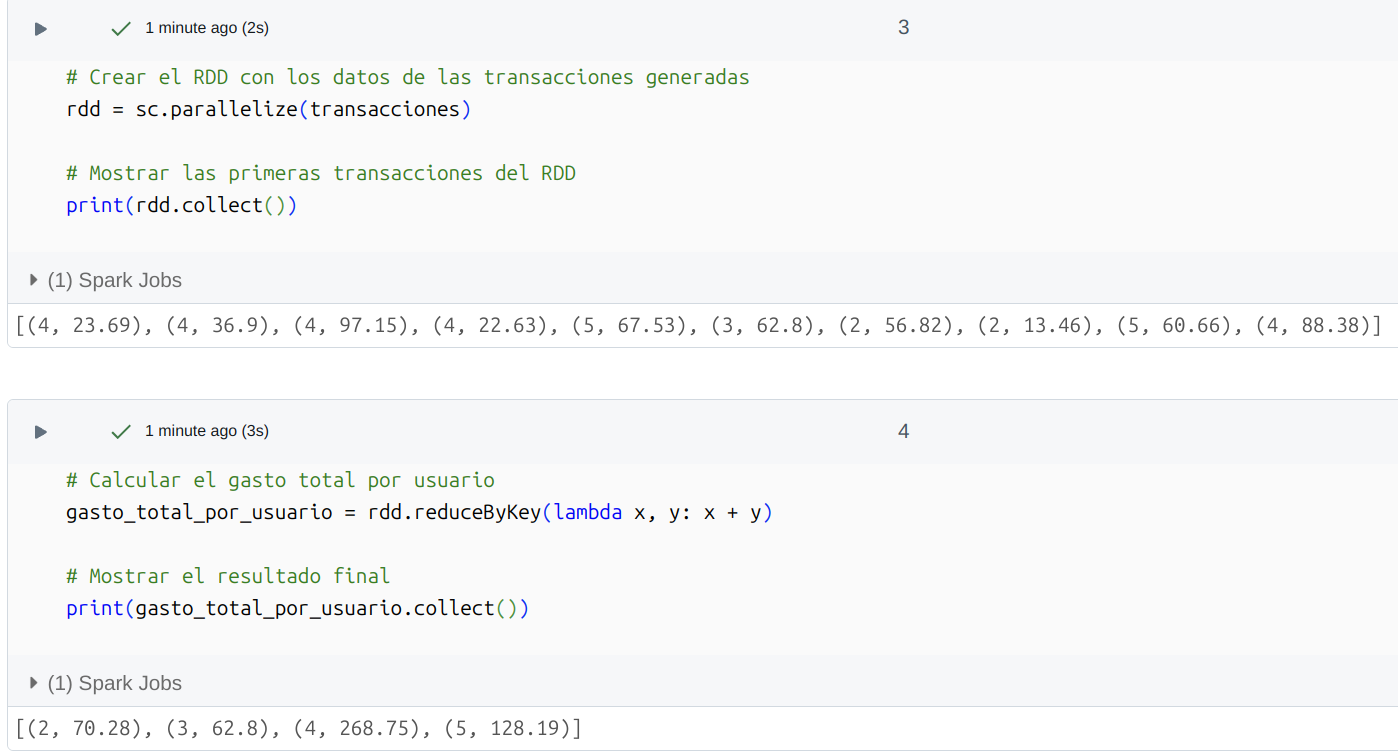


## EJERCICIOS PARA DATABRICKS

### EJERCICIO 5

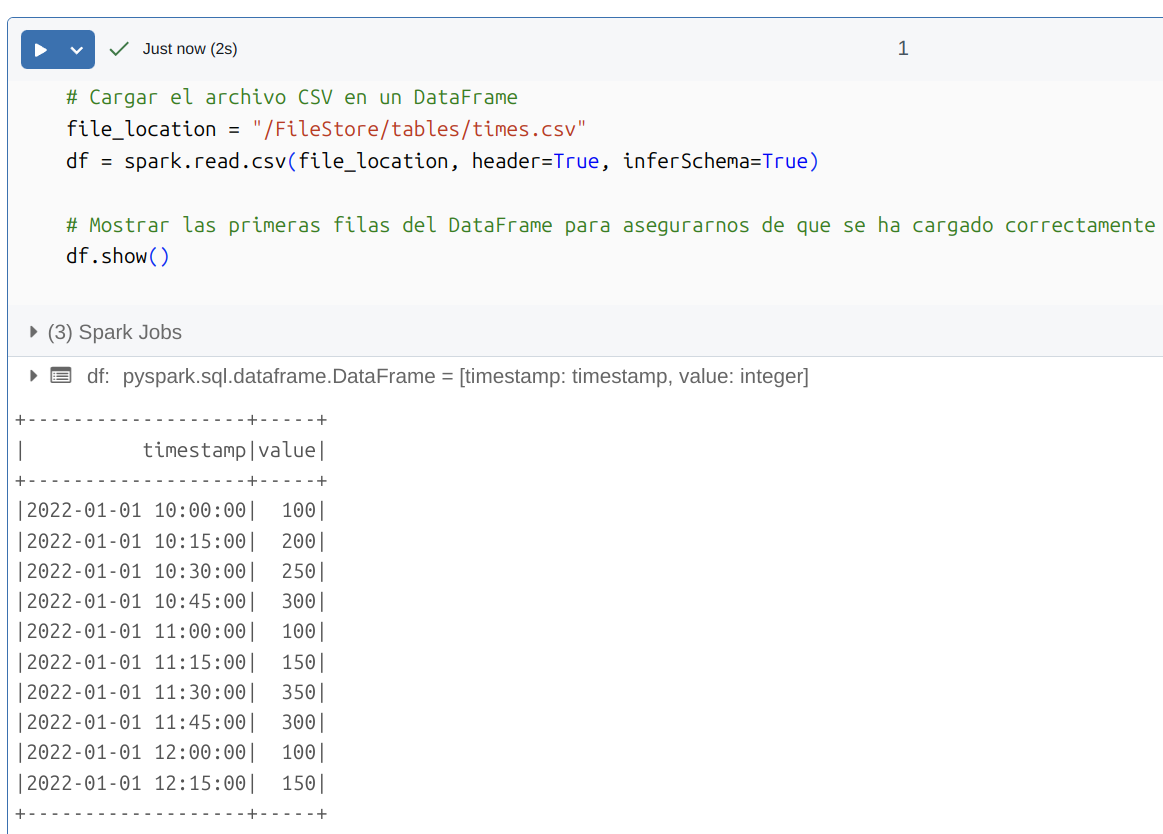
Utilizando un RDD que contiene transacciones de compra (ID de usuario, cantidad gastada), calcula el gasto total por usuario (el rdd por lo menos tiene que tener 10 líneas).

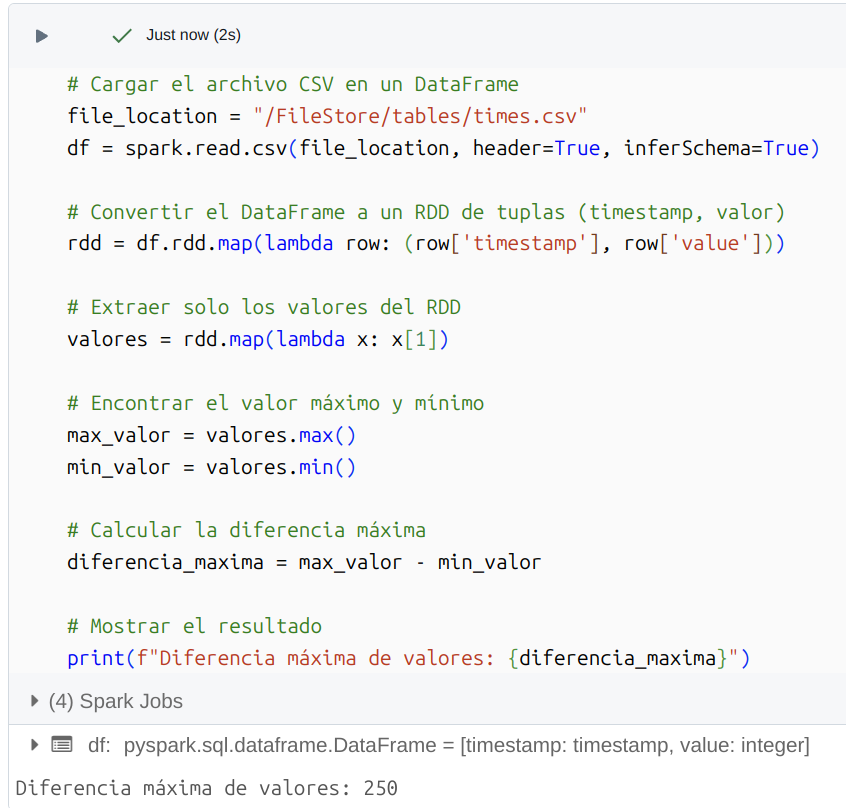




### EJERCICIO 6

Dado un RDD de valores (timestamp, valor), encuentra la diferencia máxima de valores (el rdd tienes que leerlo desde el archivo times.csv que tendrás que subir a Databricks).





### EJERCICIO 7

Implementa un algoritmo para contar la frecuencia de cada palabra en un conjunto de documentos, muestra la frecuencia de cada palabra ordenada de forma descendente.

