Ejercicios prácticos para trabajar con transformaciones y acciones en RDD , utilizando Apache Spark. Estos ejercicios ayudarán a entender cómo manipular y procesar grandes volúmenes de datos de manera distribuida.

**Transformaciones**

**map**  
**Ejercicio:**  
Dado un RDD con números, multiplica cada número por 2 y muestra el resultado.

rdd = sc.parallelize([1, 2, 3, 4, 5])

result = rdd.map(lambda x: x \* 2)

result.collect() # [2, 4, 6, 8, 10]

**filter**  
**Ejercicio:**  
Filtra todos los números pares de un RDD que contiene números del 1 al 10.

rdd = sc.parallelize([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

result = rdd.filter(lambda x: x % 2 == 0)

result.collect() # [2, 4, 6, 8, 10]

**flatMap**  
**Ejercicio:**  
Dado un RDD con listas de palabras, aplana el RDD en un solo RDD con todas las palabras.

rdd = sc.parallelize([["apple", "banana"], ["cherry", "date"], ["elderberry"]])

result = rdd.flatMap(lambda x: x)

result.collect() # ['apple', 'banana', 'cherry', 'date', 'elderberry']

**union**  
**Ejercicio:**  
Une dos RDDs que contienen diferentes conjuntos de números.

rdd1 = sc.parallelize([1, 2, 3])

rdd2 = sc.parallelize([3, 4, 5])

result = rdd1.union(rdd2)

result.collect() # [1, 2, 3, 3, 4, 5]

**intersection**  
**Ejercicio:**  
Encuentra la intersección de dos RDDs que contienen números.

rdd1 = sc.parallelize([1, 2, 3, 4])

rdd2 = sc.parallelize([3, 4, 5, 6])

result = rdd1.intersection(rdd2)

result.collect() # [3, 4]

**distinct**  
**Ejercicio:**  
Dado un RDD con valores duplicados, elimina los duplicados.

rdd = sc.parallelize([1, 1, 2, 2, 3, 3])

result = rdd.distinct()

result.collect() # [1, 2, 3]

**groupByKey**  
**Ejercicio:**  
Dado un RDD de tuplas clave-valor, agrupa los valores por clave.

rdd = sc.parallelize([('a', 1), ('b', 2), ('a', 3), ('b', 4)])

result = rdd.groupByKey().mapValues(list)

result.collect() # [('a', [1, 3]), ('b', [2, 4])]

**reduceByKey**  
**Ejercicio:**  
Suma los valores asociados a cada clave.

rdd = sc.parallelize([('a', 1), ('b', 2), ('a', 3), ('b', 4)])

result = rdd.reduceByKey(lambda x, y: x + y)

result.collect() # [('a', 4), ('b', 6)]

**sortByKey**  
**Ejercicio:**  
Ordena un RDD de tuplas clave-valor por la clave.

rdd = sc.parallelize([('b', 1), ('a', 2), ('c', 3)])

result = rdd.sortByKey()

result.collect() # [('a', 2), ('b', 1), ('c', 3)]

**join**  
**Ejercicio:**  
Realiza un join entre dos RDDs de tuplas clave-valor.

rdd1 = sc.parallelize([('a', 1), ('b', 2)])

rdd2 = sc.parallelize([('a', 3), ('b', 4)])

result = rdd1.join(rdd2)

result.collect() # [('a', (1, 3)), ('b', (2, 4))]

**cogroup**  
**Ejercicio:**  
Agrupa dos RDDs por clave.

rdd1 = sc.parallelize([('a', 1), ('b', 2)])

rdd2 = sc.parallelize([('a', 3), ('b', 4)])

result = rdd1.cogroup(rdd2)

result.collect() # [('a', ([1], [3])), ('b', ([2], [4]))]

**coalesce**  
**Ejercicio:**  
Reduzca el número de particiones de un RDD.

rdd = sc.parallelize([1, 2, 3, 4], 4)

result = rdd.coalesce(2)

print(result.getNumPartitions()) # 2

### ****Acciones****

**reduce**  
**Ejercicio:**  
Realiza una reducción para sumar todos los números de un RDD.

rdd = sc.parallelize([1, 2, 3, 4])

result = rdd.reduce(lambda x, y: x + y)

print(result) # 10

**collect**  
**Ejercicio:**  
Recoge todos los elementos de un RDD en una lista.

rdd = sc.parallelize([1, 2, 3, 4])

result = rdd.collect()

print(result) # [1, 2, 3, 4]

**count**  
 **Ejercicio:**  
 Cuenta cuántos elementos hay en un RDD.

rdd = sc.parallelize([1, 2, 3, 4])

result = rdd.count()

print(result) # 4

**first**  
 **Ejercicio:**  
 Devuelve el primer elemento del RDD.

rdd = sc.parallelize([1, 2, 3, 4])

result = rdd.first()

print(result) # 1

**take**  
 **Ejercicio:**  
 Toma los primeros 3 elementos de un RDD.

**rdd = sc.parallelize([1, 2, 3, 4])**

**result = rdd.take(3)**

**print(result) # [1, 2, 3]**

**saveAsTextFile**  
 **Ejercicio:**  
 Guarda el contenido de un RDD en un archivo de texto.

rdd = sc.parallelize([1, 2, 3, 4])

rdd.saveAsTextFile("output.txt")

**max, min**  
 **Ejercicio:**  
 Encuentra el valor máximo y mínimo de un RDD.

rdd = sc.parallelize([1, 2, 3, 4])

print(rdd.max()) # 4

print(rdd.min()) # 1

**countByKey**  
 **Ejercicio:**  
 Cuenta cuántas veces aparece cada clave en un RDD de tuplas.

rdd = sc.parallelize([('a', 1), ('b', 2), ('a', 3), ('b', 4), ('a', 5)])

result = rdd.countByKey()

print(result) # {'a': 3, 'b': 2}

**foreach**  
 **Ejercicio:**  
 Realiza una acción sobre cada elemento de un RDD, como imprimirlo.

rdd = sc.parallelize([1, 2, 3, 4])

rdd.foreach(lambda x: print(x))

Estos ejercicios cubren una amplia gama de transformaciones y acciones que puedes aplicar sobre los RDDs en Apache Spark. Son fundamentales para aprender a trabajar con datos distribuidos en sistemas de procesamiento masivo como Spark.