Ejercicio 4

Dada la siguiente lista, crea un RDD y realiza lo siguiente:

- Namelist = [("Hugo",10),("Erick",20),("Biel",30),("Antonio",15),("Manuel",25),("Francisco",15), ("Hugo",20),("Manuel",30),("Erick",20),("Hugo",10),("Francisco",30),("Erick",20),("Sofía",20), ("Biel",30),("Antonio",15),("Biel",10),("Sofía",20),("Erick",15),("Antonio",20),("Francisco",20) , ("Manuel",20),("Hugo",30)]

1. Calcula la suma de las edades de acuerdo a cada nombre usando groupbykey / reducebykey /foldbykey / combinebykey
2. Calcula el promedio de edad de cada nombre

Importaciones y creacion se SparkSession y SparkContext

In [1]:
```
import findspark
findspark.init()

import pandas as pd
import pyspark
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()
sc = spark.sparkContext
```

Punto #1

Dada la siguiente lista, crea un RDD:

- Namelist = [("Hugo",10),("Erick",20),("Biel",30),("Antonio",15),("Manuel",25),("Francisco",15), ("Hugo",20),("Manuel",30),("Erick",20),("Hugo",10),("Francisco",30),("Erick",20),("Sofía",20), ("Biel",30),("Antonio",15),("Biel",10),("Sofía",20),("Erick",15),("Antonio",20),("Francisco",20) , ("Manuel",20),("Hugo",30)]

In [2]:
```
nameList = [("Hugo",10),("Erick",20),("Biel",30),("Antonio",15),("Manuel",25),("Franci

nameRDD = sc.parallelize(nameList)
```

Punto #2

Calcula la suma de las edades de acuerdo a cada nombre usando groupbykey / reducebykey / foldbykey / combinebykey

groupByKey

In [3]:
```
nameGroupByKey = nameRDD.groupByKey().mapValues(lambda edades: sum(edades))
nameGroupByKey.collect()
```

Out[3]:
```
[('Sofía', 40),
 ('Biel', 70),
 ('Manuel', 75),
 ('Hugo', 70),
 ('Erick', 75),
 ('Antonio', 50),
 ('Francisco', 65)]
```

ReduceByKey

In [11]:
```
nameReduceByKey = nameRDD.reduceByKey(lambda value1, value2: value1 + value2)
nameReduceByKey.collect()
```

Out[11]:
```
[('Sofía', 40),
 ('Biel', 70),
 ('Manuel', 75),
 ('Hugo', 70),
 ('Erick', 75),
 ('Antonio', 50),
 ('Francisco', 65)]
```

foldByKey

In [17]:
```
nameFoldByKey = nameRDD.foldByKey(0, lambda value1, value2: value1 + value2)
nameFoldByKey.collect()
```

Out[17]:
```
[('Sofía', 40),
 ('Biel', 70),
 ('Manuel', 75),
 ('Hugo', 70),
 ('Erick', 75),
 ('Antonio', 50),
 ('Francisco', 65)]
```

combineByKey

In [27]:
```
# Función para como tratar los values
def createCombiner(value):
    return value

# Función de que hacer con los valores de una key
def mergeValues(acc, value):
    return acc + value

# Función de que hacer con cada elemento/tupla/agrupacion
def mergeCombiners(acc1, acc2):
    return acc1 + acc2

# Calcular la suma de las edades para cada nombre utilizando combineByKey()
nameSum = nameRDD.combineByKey(createCombiner, mergeValues, mergeCombiners)
```

```
# Ver los resultados
nameSum.collect()
```

Out[27]:
```
[('Sofía', 40),
 ('Biel', 70),
 ('Manuel', 75),
 ('Hugo', 70),
 ('Erick', 75),
 ('Antonio', 50),
 ('Francisco', 65)]
```

Punto 3

Calcula el promedio de edad de cada nombre

In [9]:
```
nameAvg = nameRDD.groupByKey().mapValues(lambda edades: mean(edades))
nameAvg.collect()
```

Out[9]:
```
[('Sofía', 20),
 ('Biel', 23.333333333333332),
 ('Manuel', 25),
 ('Hugo', 17.5),
 ('Erick', 18.75),
 ('Antonio', 16.666666666666668),
 ('Francisco', 21.666666666666668)]
```