# 7506-2020-2 Parcialito de Spark

Santiago Locatelli

TOTAL POINTS

**65 / 100**

QUESTION 1

## 1 Spark 65 / 100

✓ **- 5 pts** **Map innecesario**

Punto a)

✓ **- 30 pts** **Join innecesario contra Recetas_rdd**

1. Join innecesario, en ingredientes_rdd ya tenes el Id_Receta

2. *Mapea algo que ya esta bien mapeado.

3. *Reduce por nombre de receta y no por su id. ¿ Y si dos recetas tienen el mismo nombre ?

```
!pip install pyspark
!pip install -U -q PyDrive
!apt install openjdk-8-jdk-headless -qq
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.6/dist-packages (3.0.1)
Requirement already satisfied: py4j==0.10.9 in /usr/local/lib/python3.6/dist-packages (from pyspark) (0.10.9)
openjdk-8-jdk-headless is already the newest version (8u265-b01-0ubuntu2~18.04).
0 upgraded, 0 newly installed, 0 to remove and 21 not upgraded.
```

```
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from oauth2client.client import GoogleCredentials
from pyspark.sql import *
from pyspark.sql.functions import *
from pyspark import SparkContext
from pyspark.sql import SQLContext
import pandas as pd
```

```
#Creamos la sesion de Spark
spark = SparkSession.builder.getOrCreate()

#creamos el contexto de Spark
sc = spark.sparkContext
```

```
recetas = [
           (1, 'fideos con salsa', 'italiana'),
           (2, 'pollo al spiedo', 'salada'),
           (3, 'pizza cuatro quesos', 'italiana'),
           (4, 'carne asada con salsa', 'Argentino'),
           (5, 'pollo al disco', 'Mediterranea'),
           (6, 'una receta', 'Mediterranea'),
           (7, 'otra receta', 'Mediterranea'),
```

```
            (8, 'pollo al horno', 'salada'),
            (9, 'patita de pollo', 'salada'),

    ]

    ingredientes = [
            (2, 'spiedo', 2),
            (2, 'pollo', 2),
            (1, 'crema', 1),
            (1, 'queso', 0.2),
            (7, 'noEsPapa', 2),
            (7, 'noEsPollo', 1),
            (5, 'pollo', 2),
            (5, 'hongos', 0.5),
            (8, 'pollo', 2),
            (8, 'hongos', 0.5),
            (6, 'papa', 2),
            (6, 'noEsPapa', 2),
    ]


    recetas_rdd = sc.parallelize(recetas)
    ingredientes_rdd = sc.parallelize(ingredientes)
```

B)

```
    recetas_mediterraneas = recetas_rdd.filter(lambda x: x[2] == 'Mediterranea')
    recetas_con_ingredientes = recetas_mediterraneas.join(ingredientes_rdd)
    recetas_con_ingredientes.collect()
```

```
     [(5, ('pollo al disco', 'pollo')),
      (5, ('pollo al disco', 'hongos')),
      (6, ('una receta', 'papa')),
      (6, ('una receta', 'noEsPapa')),
      (7, ('otra receta', 'noEsPapa')),
      (7, ('otra receta', 'noEsPollo'))]
```

```
    def esPolloOPapa(x):
```

```
def esPolloOPapa(x):
  if( x=='pollo' or x== 'papa'):
    return 1
  else:
    return 0


sin_pollo_y_papa = recetas_con_ingredientes.map(lambda x: (x[1][0], esPolloOPapa(x[1][1]) ) )
sin_pollo_y_papa.take(5)
```

```
[('pollo al disco', 1),
 ('pollo al disco', 0),
 ('una receta', 1),
 ('una receta', 0),
 ('otra receta', 0)]
```

```
recetas_finales = sin_pollo_y_papa.map(lambda x: (x[0],x[1])).reduceByKey(lambda x,y: x + y ).filter(lambda x: x[1] == 0)
recetas_finales.take(5)
```

```
[('otra receta', 0)]
```

A)

```
usan_pollo = ingredientes_rdd.filter(lambda x: x[1] == 'pollo').join(recetas_rdd).map(lambda x: (x[0], x[1][1]))
usan_pollo.take(5)
```

```
[(8, 'pollo al horno'), (5, 'pollo al disco'), (2, 'pollo al spiedo')]
```

```
ingredientes_y_pollo = usan_pollo.join( ingredientes_rdd ).map(lambda x: (x[1][1], 1 ) ).filter(lambda x: x[0] != 'pollo')\
.reduceByKey(lambda x,y: x + y)
ingredientes_y_pollo.take(5)
```

```
[('hongos', 2), ('spiedo', 1)]
```

## 1 Spark 65 / 100

✓ **- 5 pts** **Map innecesario**

Punto a)

✓ **- 30 pts** **Join innecesario contra Recetas_rdd**

1. Join innecesario, en ingredientes_rdd ya tenes el Id_Receta
2. *Mapea algo que ya esta bien mapeado.
3. *Reduce por nombre de receta y no por su id. ¿ Y si dos recetas tienen el mismo nombre ?