7506-2021 Spark - 2da Oportunidad

Román Vázquez Lareu

TOTAL POINTS

65 / 100

QUESTION 1

1 Spark 65 / 100

✓ **- 20 pts No cachea RDD con tests del primer semestre.**

Punto a

✓ **- 5 pts Calcula cantidades por localidad innecesariamente.**

Punto b

✓ **- 10 pts Falta acción.**

1. Esto deberias cachearlo para usarlo en el punto b.
2. (x[3],1)
3. No es necesario calcular esto por localidad.
4. SortByKey es una transformación, falta acción.

```
# (fecha, dni, id_localidad, resultado)
#fecha: YYYY-MM-DD (primer semestre mes 01,02,03,04,05,06)
tests = [
         ("2021-03-01",123,1,1),
         ("2021-03-01",123,1,1),
         ("2021-03-01",122,1,0),
         ("2021-03-01",122,2,1),
         ("2021-02-01",142,3,0),
         ("2021-01-01",172,4,1),
         ("2021-01-01",172,4,0),
         ("2021-01-01",192,4,1),
         ("2021-01-01",192,5,0),
         ("2021-02-01",222,7,0),
         ("2021-02-01",222,7,1),
         ("2021-01-01",222,8,1),
         ("2021-04-01",222,8,1),
         ("2021-04-01",222,9,0),
         ("2021-04-01",222,9,0),
         ("2020-07-01",222,9,0)

]
# (id_localidad, nombre, provincia)

localidades = [
               (1,"nombre1","Cordoba"),
               (2,"nombre2","Cordoba"),
               (3,"nombre3","Cordoba"),

               (4,"nombre4","Formosa"),
               (5,"nombre5","Formosa"),

               (6,"nombre6","Santa fe"),
               (7,"nombre7","Santa fe"),
               (8,"nombre8","Santa fe"),
               (9,"nombre9","Santa fe"),

]

testsRdd = sc.parallelize(tests)
localidadesRdd = sc.parallelize(localidades)
```

Indicar la provincia con mayor porcentaje de tests positivos en el primer semestre de 2021, teniendo en cuenta sólo las provincias con al menos 100 tests en dicho trimestre.

```
#tests del primer semestre
testsSemestrePorLocalidad = testsRdd.filter(lambda x: x[0].split("-")[1] in ["01","02","03","04","05","06"] and x[0].split("-")[0] == "2021")\
                            .map(lambda x: (x[2],(1,1) if x[3] == 1 else (0,1))).reduceByKey(lambda x,y: (x[0]+y[0],x[1]+y[1]))
#(id_localidad),(positivos,#tests totales)
```

1 2 3

```
[(2, (1, 1)),
 (4, (2, 3)),
 (8, (2, 2)),
 (1, (2, 3)),
 (3, (0, 1)),
 (5, (0, 1)),
 (7, (1, 2)),
 (9, (0, 2))]
```

```
localidadesTojoin = localidadesRdd.map(lambda x: (x[0],x[2]))
```

```
[(1, 'Cordoba'),
 (2, 'Cordoba'),
 (3, 'Cordoba'),
 (4, 'Formosa'),
 (5, 'Formosa'),
 (6, 'Santa fe'),
 (7, 'Santa fe'),
 (8, 'Santa fe'),
 (9, 'Santa fe')]
```

```
# filtro por 4 tests por el tamaño del set de datos, iria 100
testsSemestrePorLocalidad.join(localidadesTojoin).map(lambda x: (x[1][1],x[1][0])).reduceByKey(lambda x,y: (x[0]+y[0],x[1]+y[1]))\
                  .filter(lambda x: x[1][1]>4).map(lambda x: (x[0],x[1][0]/x[1][1]) ).reduce(lambda x,y: x if x[1]>y[1] else y)[0]
```

```
'Cordoba'
```

Indicar cantidad de localidades por rango de porcentaje de tests positivos en el primer semestre de 2021 (utilizando rangos de 10%). El resultado debería tener la siguiente estructura:

```
#los intervalos los tomo:
#  [0,10%) -> 0
#  [10%,20%) -> 1
#  [20%,30) -> 2
#  [30%,40) -> 3
#  [40%,50) -> 4
#  [50%,60) -> 5
#  [60%,70) -> 6
#  [70%,80) -> 7
#  [80%,90) -> 8
# (90,100] -> 9

# => estructura:  (rango inferior,cantidad de localidades)
#el if es para que el caso 100% quede en el bucket que le corresponde (el 9)
testsSemestrePorLocalidadPositividad = testsSemestrePorLocalidad.map(lambda x: (int(x[1][0]/x[1][1] * 10),1) if int(x[1][0]/x[1][1] * 10) != 10 else (9,1) )\
                                    .reduceByKey(lambda x,y: x+y).sortByKey()
```

```
[(0, 3), (5, 1), (6, 2), (9, 2)]
```

1 Spark 65 / 100

✓ **- 20 pts** **No cachea RDD con tests del primer semestre.**

Punto a

✓ **- 5 pts** **Calcula cantidades por localidad innecesariamente.**

Punto b

✓ **- 10 pts** **Falta acción.**

1. Esto deberias cachearlo para usarlo en el punto b.
2. (x[3],1)
3. No es necesario calcular esto por localidad.
4. SortByKey es una transformación, falta acción.