Universidad Internacional de Valencia

Master en Big Data y Data Science

Problema 3: Persona y método de pago.

Procesamiento de datos masivos: Spark

Autor:
Adrián Hernández Padrón
Julio 2022

Universidad Internacional de Valencia

1. Código

Una vez se han leido los datos de entrada, vamos a mapearlos. Para ello creamos dos funciones, una que se encargue del saldo mayor de 1500 y otra que se encargue del saldo menor o igual a 1500. Como solo evaluamos el saldo en tarjetas de crédito tenemos que poner un condicional para que agregue solamente las compras con tarjeta de crédito y también para cubrir el caso de cualquier persona que solo tenga compras sin tarjeta de crédito.

El primer paso será dividir por línea la entrada y después, con un bucle, separar cada línea del fichero de entrada. Aplicamos el condicional y lo añadimos al array vacío en cada caso para tener la información correctamente mapeada con la clave-valor: nombre-1, de manera que podamos sumar luego los contadores.

```
def mapMayor(linea):
    new_linea = linea.split("\n")
    for element in new_linea:
        mayor_counter = []
        nombre, metodo , saldo = element.split(";")
        if metodo == 'Tarjeta de crédito':
            if int(saldo) > 1500:
                mayor_counter.append((nombre, 1))
        else:
            mayor_counter.append((nombre, 0))
    return mayor_counter

def mapMenor(linea):
    new_linea = linea.split("\n")
    for element in new_linea:
        menor_counter = []
        nombre, metodo , saldo = element.split(";")
        if metodo == 'Tarjeta de crédito':
            if int(saldo) <= 1500:
                menor_counter.append((nombre, 1))
        else:
            menor_counter.append((nombre, 0))
    return menor_counter
```

Figura 1: El else se encarga de cubrir el caso de que una persona exista en el fichero pero no tenga compras con tarjeta de crédito.

Ahora usamos el flatmap y el reduceByKey para resolver el ejercicio, lanzando uno para las compras superiores a 1500 y otro para las inferiores. También nos ayudamos de un map para escribir los resultados con la estructura deseada.

```
mayor = datosEntrada.flatMap(mapMayor).reduceByKey(lambda x, y: x +y)
resultadomayor = mayor.map(lambda linea: linea[0]+";"+str(linea[1])).repartition(1)
#Calculamos con un reduceByKey y la función anterior cuantas veces han gastado menos
menor = datosEntrada.flatMap(mapMenor).reduceByKey(lambda x, y: x +y)
resultadomenor = menor.map(lambda linea: linea[0]+";"+str(linea[1])).repartition(1)
```

Y ya por último guardamos la salida en dos carpetas distintas.

```
#Guardamos en las dos carpetas
resultadomayor.saveAsTextFile(salida1)
resultadomenor.saveAsTextFile(salida2)
```

2. Ejecución y resultados

Para lanzar el programa en este caso, escribimos lo siguiente por la linea de comandos:
$spark-submit personaYMetodosDePago.py file$: $/Users/adrihp/Master/MBID03/scriptsSpark/Problema3/e$
$/Users/adrihp/Master/MBID03/scriptsSpark/Problema3/comprasCreditoMayorDe1500 file$:
$/Users/adrihp/Master/MBID03/scriptsSpark/Problema3/comprasCreditoMenorDe1500$

en donde las últimas dos rutas hacen referencia a las dos carpetas en donde vamos a guardar la salida.

La entrada y la salida del programa fueron las siguientes.

```
Alice;Tarjeta de crédito;1000
Alice;Tarjeta de crédito;1800
Alice;Tarjeta de crédito;2100
Bob;Bizum;2000
```

```
(base) adrihp@MacBook-Air-de-Adrian-2 Problema3 % cat comprasCreditoMayorDe1500/part-00000
Alice;2
Bob;0
(base) adrihp@MacBook-Air-de-Adrian-2 Problema3 % cat comprasCreditoMenorDe1500/part-00000
Alice;1
Bob;0
```