Alumno: **Bru Montes, Israel**

Edición **Octubre 2024 – Grupo A** a 26/12/2024

# **1. (6 ptos) Dado un dataset que contenga entradas con la forma “persona;método\_pago;dinero\_gastado”, crea un programa llamado personaGastosConTarjetaCredito que para cada persona indique la suma del dinero gastado con tarjeta de crédito, con el formato persona;gastoconTDC. Ejemplo:**

**Entrada Salida**

**Alice;Tarjeta de crédito;100 Alice;250**

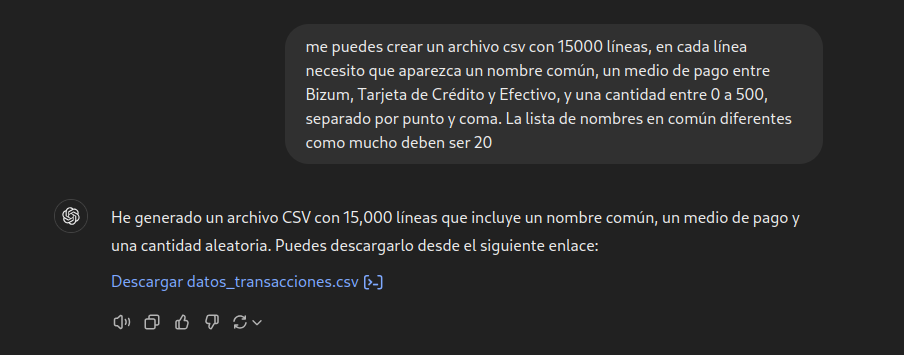
**Alice;Tarjeta de crédito;150 Bob;201**

**Alice;Bizum;200 Luis;0**

**Bob;Tarjeta de crédito;201**

**Luis;Bizum;300**

Generamos con ChatGPT un archivo csv para probar con las condiciones que se muestran en la siguiente captura:



La estrategia sería leer con el metodo “textFile” la carpeta donde se ubican todos los archivos .txt con los datos. Para procesar posteriormente:

1. Hacemos un split utilizando como carácter de separación “;” para cada elemento y procesamos por cada línea el archivo de entrada.

2. Utilizamos “filter” para filtrar tanto las líneas del archivo que no contengan al menos en el split 2 elementos para evitar accesos a elementos del split que no existen y filtramos por “Tarjeta de Crédito”.

3. Usamos map() para elegir las columnas a procesar, en la posición 3 (categoría) y la 5 (número de visitas) para generar una tupla con los datos necesarios para tratar en la siguiente iteración.

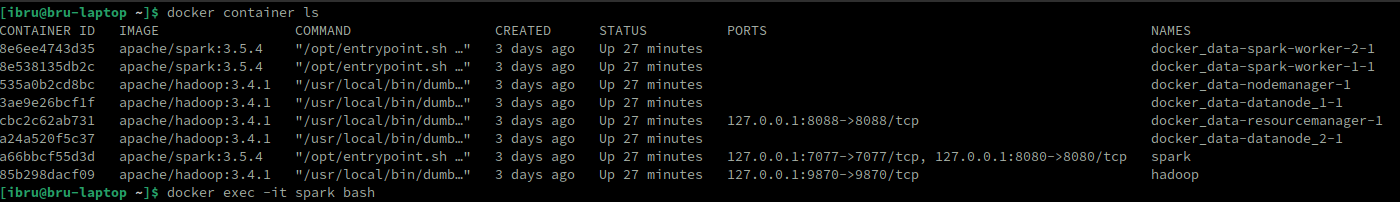
4. Mediante el método ReduceByKey sumamos el número total de visitas de la categoria e invocamos el método orderBy para ordenar por número total de visitas por categoria (ascendente).

5. Usamos el método first() para obtener la primera fila del RDD, y con join convertimos la tupla obtenida en un string para imprimirlo en el fichero de salida.

Nos conectamos a la máquina de spark:

[ibru@bru-laptop ~]$ docker container ls

[ibru@bru-laptop ~]$ docker exec -it spark bash



[ibru@bru-laptop ~]$ docker exec -it spark bash

spark@spark:/opt/spark/work-dir$ ls -l

total 916

drwx------ 2 1000 1000 4096 Dec 26 15:51 0303

-rw-r--r-- 1 1000 1000 1842 Dec 27 15:58 categoriaDeVideosMenosVista.py

drwxr-xr-x 2 root root 4096 Dec 24 06:38 out.txt

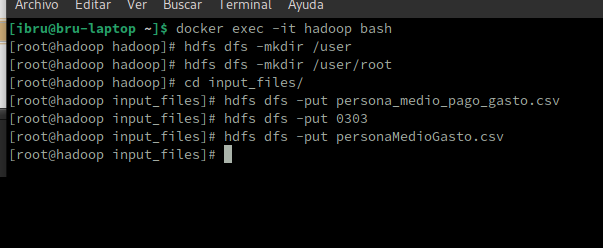
-rw-r--r-- 1 1000 1000 1102 Dec 27 15:58 personaGastosConTarjetaCredito.py

-rw-r--r-- 1 1000 1000 527306 Dec 27 16:14 personaMedioGasto.csv

-rw-r--r-- 1 1000 1000 386765 Dec 24 17:48 persona\_medio\_pago\_gasto.csv

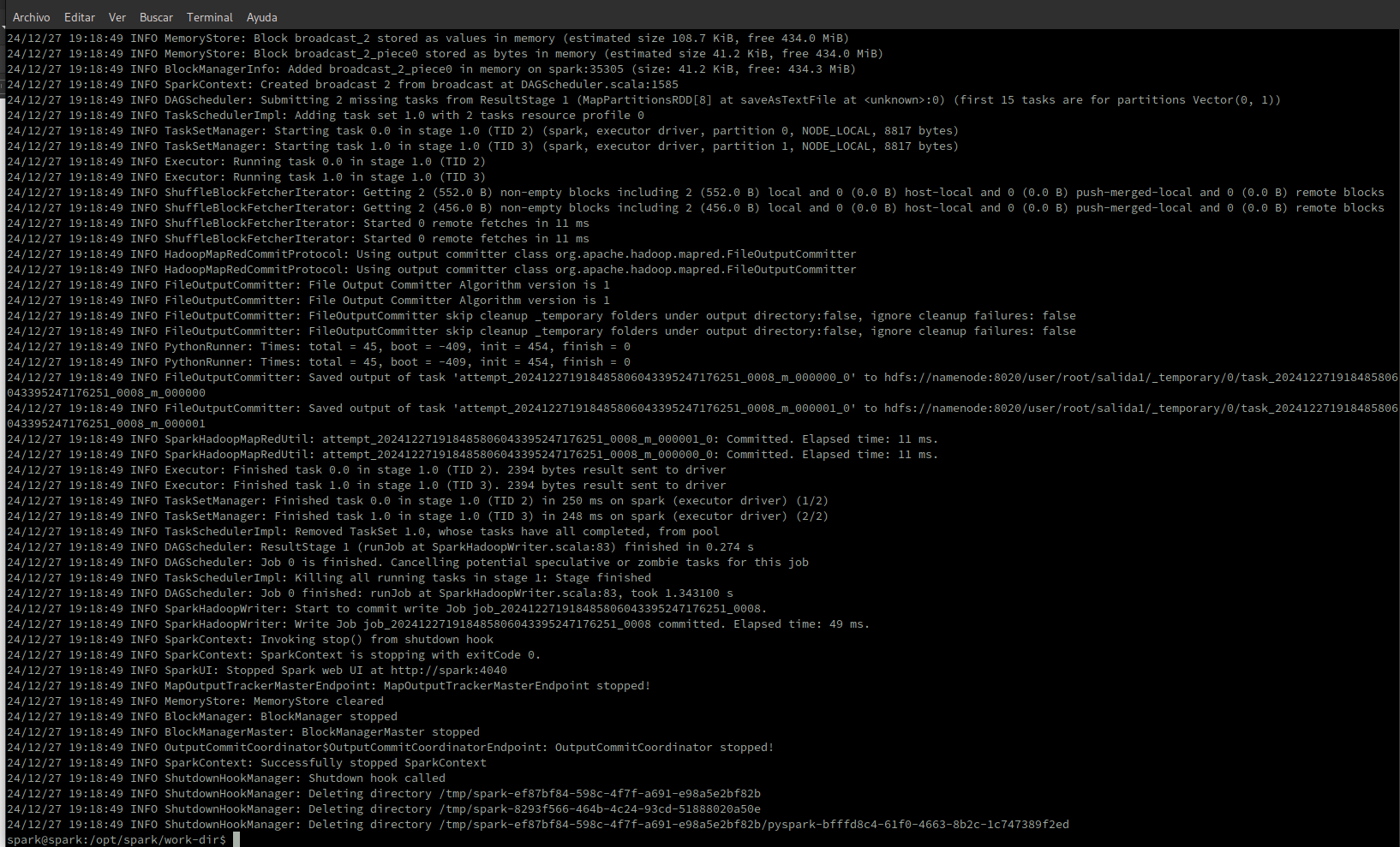
-rw-r--r-- 1 1000 1000 1830 Dec 27 18:41 personaYMetodosDePago.py

Subimos el archivo de entrada al hdfs:



Ejecutamos el script:

spark@spark:/opt/spark/work-dir$ ../bin/spark-submit personaGastosConTarjetaCredito.py persona\_medio\_pago\_gasto.csv salida1



Nos bajamos la carpeta salida1 del hdfs y examinamos un archivo de salida de la partición 0000:

[root@hadoop input\_files]# hdfs dfs -get salida1

[root@hadoop input\_files]# cat salida1/part-0000

part-00000 part-00001

[root@hadoop input\_files]# cat salida1/part-00000 |more

('Carlos', 57841.78)

('Miguel', 68878.05)

('Elena', 60661.26)

('Raúl', 64781.31)

('Roberto', 72557.89)

('Carmen', 63748.75)

('Antonio', 58064.83)

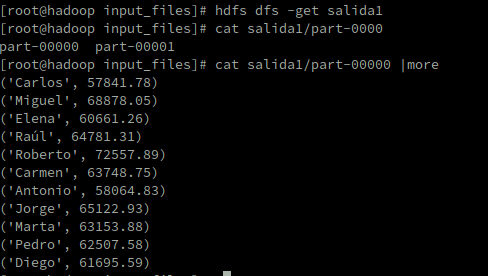
('Jorge', 65122.93)

('Marta', 63153.88)

('Pedro', 62507.58)

('Diego', 61695.59)

[root@hadoop input\_files]#



**2. (7 ptos) Dado un dataset que contiene información sobre los videos de Youtube (https://netsg.cs.sfu.ca/youtubedata/), crear un programa llamado CategoriaDeVideosMenosVista que obtenga cuál es la categoría de videos menos vista de la plataforma Youtube y el número total de visualizaciones que hay en esa categoría. El programa debe recibir dos parámetros de entrada: la carpeta en la que está el dataset y la carpeta en la que se guardará el resultado. En la carpeta donde está el dataset se tienen que descomprimir UNO de los archivos 0222.zip, 0301.zip, etc., que se encuentran en el enlace anterior.**

La estrategia sería leer con el metodo “spark.sparkContext.textFile” la carpeta donde se ubican todos los archivos .txt con los datos (He eliminado a mano el fichero log.txt). Para procesar posteriormente:

1. Hacemos un split utilizando los “tabs” o tabulaciones para separar elementos, leemos por cada línea.

2. Utilizamos “filter” para filtrar tanto las líneas del archivo que no contengan al menos en el split 5 elementos para evitar accesos a elementos del split que no existen.

3. Elegimos la columna 3 (categoría) y la 5 (número de visitas) para generar un nuevo RDD reducido con una tupla de datos necesarios para tratar en la siguiente iteración.

4. Usamos el método ReduceByKey para hacer la suma total de visitas de la categoria e invocamos el método orderBy para ordenar por número total de visitas por categoria

5. Usamos el método first para obtener la primera fila del RDD, y con join convertimos la tupla obtenida en un string para imprimirlo en el fichero de salida.

Subimos la carpeta 0303 de entrada al hdfs:

[root@hadoop input\_files]# hdfs dfs -put 0303

Comprobamos que se han subido todos los archivos:

[root@hadoop input\_files]# hdfs dfs -ls /user/root/0303

Found 4 items

-rw-r--r-- 3 root supergroup 48398 2024-12-27 19:16 /user/root/0303/0.txt

-rw-r--r-- 3 root supergroup 408822 2024-12-27 19:16 /user/root/0303/1.txt

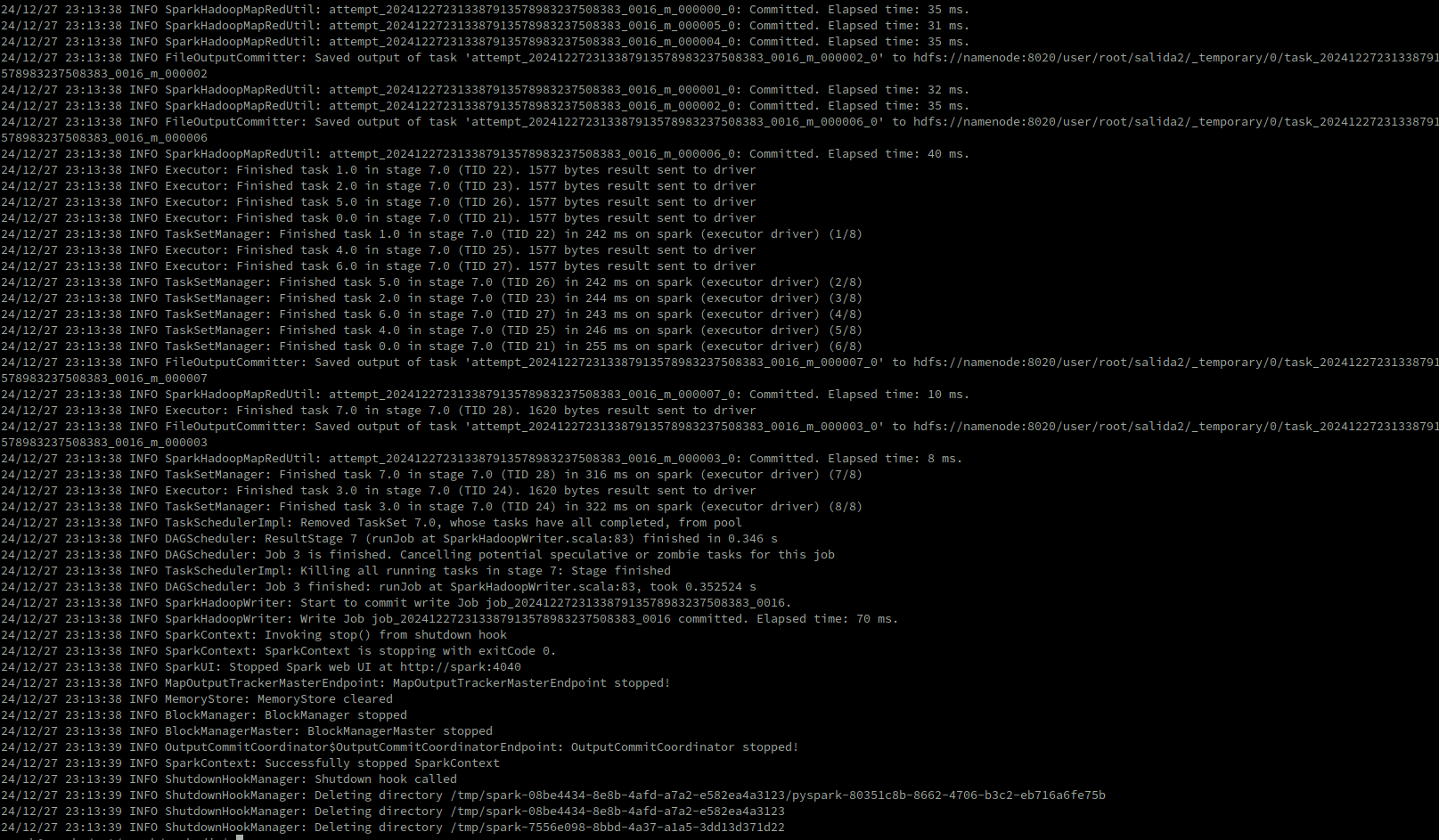
-rw-r--r-- 3 root supergroup 2479639 2024-12-27 19:16 /user/root/0303/2.txt

-rw-r--r-- 3 root supergroup 16540108 2024-12-27 19:16 /user/root/0303/3.txt

[root@hadoop input\_files]#

Ejecutamos el script:

spark@spark:/opt/spark/work-dir$ ../bin/spark-submit categoriaDeVideosMenosVista.py 0303 salida2

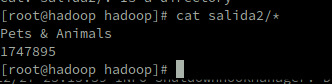


Nos bajamos del hdfs los archivos generados:

[root@hadoop hadoop]# hdfs dfs -rm -r /user/root/salida2

Y listamos los resultados:

[root@hadoop hadoop]# cat salida2/\*



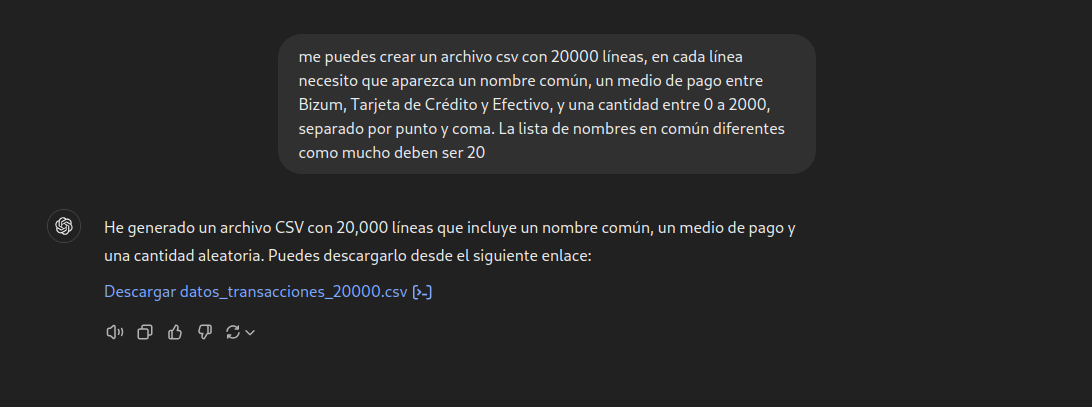
**3. (7 ptos) Dado un dataset que contenga entradas con la forma “persona;método\_pago;dinero\_gastado”, crea un programa llamado personaYMetodosDePago que:**

**a) Por cada persona indique en cuántas compras pagó más de 1500 euros con un medio de pago diferente a tarjeta de crédito. La solución se tiene que guardar en un archivo llamado comprasSinTDCMayorDe1500.**

**b) Por cada persona indique en cuántas compras pagó menos o igual a 1500 euros con un medio de pago diferente a tarjeta de crédito. La solución se tiene que guardar en un archivo llamado comprasSinTDCMenoroIgualDe1500.**

**Se valorará positivamente la eficiencia del programa, por ejemplo no usar transformaciones innecesarias.**

Generamos como en actividades anteriores un csv con datos para tratar:



En este ejercicio la estrategia es leer el archivo que nos pasan como parámetro, y una vez procesado, generaremos 2 archivos que ya tenemos predefinidos por el ejercicio. Procesaremos el archivo de entrada de la siguiente forma:

1. Hacemos un split utilizando como elemento de separación “;” por cada línea

2. Utilizamos “filter” para filtrar, tanto las líneas del archivo que contengan al menos en el split 2 elementos como para eliminar del RDD a tratar aquellos que no tienen como método de pago “Tarjeta de Crédito”.

3. El siguiente paso será fraccionar el RDD en 2 partes, según nos pide el enunciado, por una parte filtramos el RDD por gastos mayores de 1500 y contamos con countByKey()

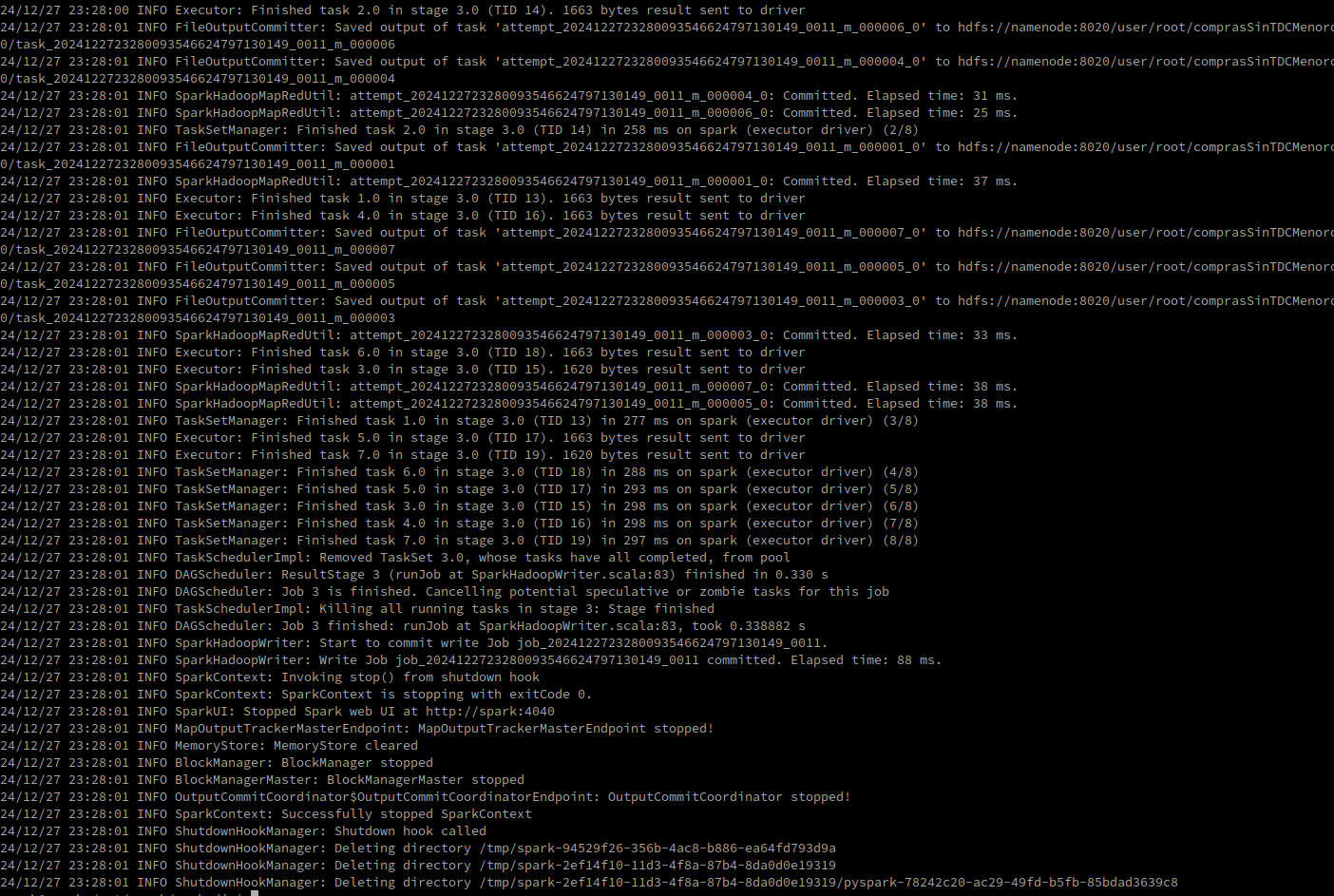
4. Imprimos los resultados en el archivo comprasSinTDCMayorDe1500.txt

5. Partiendo del RDD sin filtrar por cantidades de gasto, filtramos aquellas personas que han hecho un gasto menor o igual que 1500 y contamos con countByKey()

6. Imprimimos los resultados por líneas persona;gasto en el archivo comprasSinTDCMenoroIgualDe1500.txt

Subimos al hdfs el archivo que vamos a utilizar como entrada:

[root@hadoop hadoop]# hdfs dfs -put input\_files/personaMedioGasto.csv



Nos dirigimos al hdfs y nos bajamos los 2 archivos de salida:

[root@hadoop hadoop]# hdfs dfs -get comprasSinTDCMayorDe1500.txt

[root@hadoop hadoop]# hdfs dfs -get comprasSinTDCMenoroIgualDe1500.txt