Problema 1

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	

Notar que Alice gasta en total 450 euros, pero sólo 250 son con tarjeta de crédito (100 + 150). Se valorará positivamente la eficiencia del programa, por ejemplo, no usar transformaciones innecesarias.

1.1 Solución

- Se Inicializa creando el contexto spark y seteando ando variables para el input y output de la data. (Se debe cambiar según ejecución)

```
from pyspark import SparkContext, SparkConf

# Crear un contexto de Spark
conf = SparkConf().setAppName("problema1")
sc = SparkContext(conf=conf)

# Constantes -- cambiar según requerimiento
PATH_INPUT = './problema1/casoDePrueba1.txt'
PATH_OUT = './problema1/output/personaGastosConTarjetaCredito'
```

- Se inicializa programa con función main, generando una lectura de los datos mediante un textFile
- Aplicando una función map separamos la información según el separador de archivo ';'
- Seleccionamos columnas y damos formato a columna de gasto
- Empleamos una función map para seleccionar la persona y gasto con TDC
- Finalmente empleamos función reduceByKey para agrupar datos.

```
# 1. lectura de datos
    input_rdd = sc.textFile(PATH_INPUT)

    # 2. Procesamiento de datos
    # 2.1 Convirtiendo datos en tupla
    tuple_rdd = (input_rdd
        .map(lambda line: line.split(";"))
        .map(lambda x: (x[0], x[1], float(x[2])))
    )
    # 2.2 Seleccionamos persona y gasto en caso este sea con TDC en otro
caso se coloca 0
    process_rdd = tuple_rdd.map(lambda x: (x[0], x[2] if x[1] == 'Tarjeta
de crédito' else 0))

    # 2.3 Sumarizamos los gastos por persona
    reduce_rdd = process_rdd.reduceByKey(lambda x, y: x + y)
```

- Por ultimo se genera función de escritura para almacenar datos en output path.

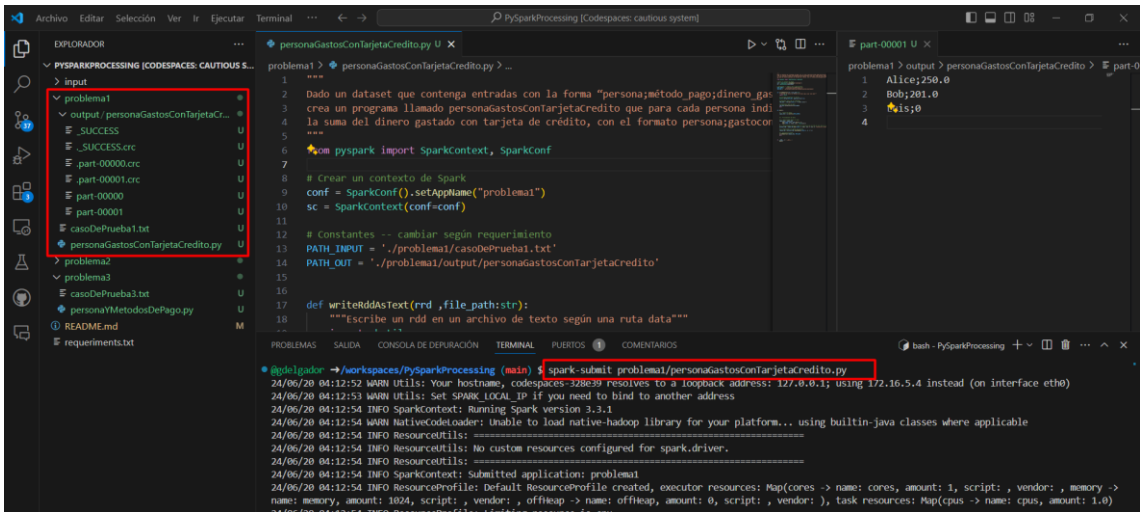
```
def writeRddAsText(rrd ,file_path:str):
    """Escribe un rdd en un archivo de texto según una ruta data"""
    import shutil

    # Eliminamos directorio si existe
    shutil.rmtree(file_path, ignore_errors=True)

    # Generamos nuevo rdd con separador
    rdd_save = rrd.map(lambda x: f"{x[0]};{x[1]}")

    # almacenamos data en path
    rdd_save.saveAsTextFile(file_path)
    pass
```

1.2 Ejecución de Programa

Se ejempla comando de ejecución desde el repositorio

```
spark-submit problema1/personaGastosConTarjetaCredito.py
```

1.2 Evidencia Ejecución


EXPLORADOR
PYSPARKPROCESSING [CODESPACES: CAUTIOUS S...
input
problema1
output / personaGastosConTarjetaCr...
_SUCCESS
_SUCCESS.crc
.part-00000.crc
.part-00001.crc
part-00000
part-00001
casoDePrueba1.txt
personaGastosConTarjetaCredito.py
problema2
problema3
casoDePrueba3.txt
personaYMetodosDePago.py
README.md
requeriments.txt
personaGastosConTarjetaCredito.py U
problema1 > personaGastosConTarjetaCredito.py > ...
Dado un dataset que contenga entradas con la forma "persona;método_pago;dinero_gas
crea un programa llamado personaGastosConTarjetaCredito que para cada persona indi
la suma del dinero gastado con tarjeta de crédito, con el formato persona;gastocor
from pyspark import SparkContext, SparkConf
# Crear un contexto de Spark
conf = SparkConf().setAppName("problema1")
sc = SparkContext(conf=conf)
# Constantes -- cambiar según requerimiento
PATH_INPUT = './problema1/casoDePrueba1.txt'
PATH_OUT = './problema1/output/personaGastosConTarjetaCredito'
def writeRddAsText(rrd ,file_path:str):
"""Escribe un rdd en un archivo de texto según una ruta data"""
part-00001 U
problema1 > output > personaGastosConTarjetaCredito > part-0
Alice;250.0
Bob;201.0
is;0
PROBLEMAS
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL
PUERTOS
COMENTARIOS
bash - PySparkProcessing
@gdelgador ➜ /workspaces/PySparkProcessing (main) $ spark-submit problema1/personaGastosConTarjetaCredito.py
24/06/20 04:12:52 WARN Utils: Your hostname, codespaces-328e39 resolves to a loopback address: 127.0.0.1; using 172.16.5.4 instead (on interface eth0)
24/06/20 04:12:53 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
24/06/20 04:12:54 INFO SparkContext: Running Spark version 3.3.1
24/06/20 04:12:54 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
24/06/20 04:12:54 INFO ResourceUtils: ==========
24/06/20 04:12:54 INFO ResourceUtils: No custom resources configured for spark.driver.
24/06/20 04:12:54 INFO ResourceUtils: ==========
24/06/20 04:12:54 INFO SparkContext: Submitted application: problema1
24/06/20 04:12:54 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script: , vendor: , memory ->
name: memory, amount: 1024, script: , vendor: , offHeap -> name: offHeap, amount: 0, script: , vendor:), task resources: Map(cpus -> name: cpus, amount: 1.0)