

Asignatura	Datos del alumno	Fecha
Ingeniería para el Procesado Masivo de Datos	Apellidos: Castro Méndez	13 febrero 2022
	Nombre: Nikolai	

```

spartan-ripple-338700 > datanikocluster
Sign out

File Edit View Run Kernel Tabs Settings Help

Launcher x actividad_2.ipyn x
+ - Code PySpark C

Actividad 2: Structured Streaming y Kafka

Recuerda borrar siempre las líneas que dicen raise NotImplementedError

Punto de partida (final de la actividad 1): función retrasoMedio

Para los vuelos que llegan con retraso positivo, calcular para cada aeropuerto de llegada el retraso medio.

Recordatorio: El código que calcule esto debería ir encapsulado en una función de Python que reciba como argumento un DataFrame y devuelva como resultado el DataFrame con el cálculo del retraso medio por aeropuerto, ordenado de mayor a menor retraso medio. La columna creada con el retraso medio debe llamarse retraso_medio.

Copia en la siguiente celda el código de tu función retrasoMedio que has completado en la actividad 1. El DataFrame devuelto por la función debería tener solamente dos columnas: dest y retraso_medio.

In [1]: def retrasoMedio(df):
        c_restra_medio = df.filter(F.col("arr_delay") > 0)\
            .groupBy("dest")\
            .agg(F.mean("arr_delay").alias("retraso_medio"))\
            .sort("retraso_medio", ascending = False)

        return c_restra_medio

In [2]: # Reemplaza por el código correcto siguiendo las indicaciones anteriores
retrasosStreamingDF = spark.readStream\
    .format("kafka")\
    .option("kafka.bootstrap.servers", "datanikocluster-w-1:9092")\
    .option("subscribe", "retrasos")\
    .load()

In [3]: # Mostramos el esquema de este DataFrame
types = retrasosStreamingDF.dtypes
assert(retrasosStreamingDF.isStreaming)
assert((types[0][0] == "key") & (types[0][1] == "binary"))
assert((types[1][0] == "value") & (types[1][1] == "binary"))
assert((types[2][0] == "topic") & (types[2][1] == "string"))
assert((types[3][0] == "partition") & (types[3][1] == "int"))
assert((types[4][0] == "offset") & (types[4][1] == "bigint"))
assert((types[5][0] == "timestamp") & (types[5][1] == "timestamp"))
assert((types[6][0] == "timestampType") & (types[6][1] == "int"))

In [4]: retrasosStreamingDF.printSchema()

root
 |-- key: binary (nullable = true)
 |-- value: binary (nullable = true)
 |-- topic: string (nullable = true)
 |-- partition: integer (nullable = true)
 |-- offset: long (nullable = true)
 |-- timestamp: timestamp (nullable = true)
 |-- timestampType: integer (nullable = true)

In [5]: from pyspark.sql.types import StructType, StructField, StringType, DoubleType
from pyspark.sql import functions as F

esquema = StructType([
    StructField("dest", StringType()),
    StructField("arr_delay", DoubleType())
])

parsedDF = retrasosStreamingDF\
    .select("value")\
    .withColumn("value", F.col("value").cast(StringType()))\
    .withColumn("parejas", F.from_json(F.col("value"), esquema))\
    .withColumn("dest", F.col("parejas.dest"))\
    .withColumn("arr_delay", F.col("parejas.arr_delay"))

In [6]: tipos = parsedDF.dtypes
assert(("value", "string") in tipos)
assert(("parejas", 'struct<dest:string,arr_delay:double>') in tipos)
assert(("dest", 'string') in tipos)
assert(("arr_delay", 'double') in tipos)

In [7]: # Evalúa el siguiente código pero no lo modifiques
# Indicamos que este DataFrame se guarde en memoria cuando se va actualizando,
# y arrancamos la ejecución en Streaming con la acción start()

retrasoMedioStreamingDF = retrasoMedio(parsedDF)

consoleOutput = retrasoMedioStreamingDF\
    .writeStream\
    .queryName("retrasosAgg")\
    .outputMode("complete")\
    .format("memory")\
    .start()

```

Asignatura	Datos del alumno	Fecha
Ingeniería para el Procesado Masivo de Datos	Apellidos: Castro Méndez	13 febrero 2022
	Nombre: Nikolai	

```

Linux datanikocluster-m 5.10.0-0-bpo.9-amd64 #1 SMP Debian 5.10.70-1-bpo10+1 (2021-10-10) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
nickcm08@datanikocluster-m:~$ /usr/lib/kafka/bin/kafka-topics.sh --zookeeper localhost:2181 --create
--replication-factor 1 --partitions 1 --topic retrasos
Created topic "retrasos".
nickcm08@datanikocluster-m:~$ /usr/lib/kafka/bin/kafka-topics.sh --zookeeper localhost:2181 --list
retrasos
nickcm08@datanikocluster-m:~$ /usr/lib/kafka/bin/kafka-console-producer.sh --broker-list datanikoclu
ster-w-0:9092 --topic retrasos
>{"dest": "GRX", "arr_delay": 2.6}
{"dest": "MAD", "arr_delay": 5.4}
{"dest": "GRX", "arr_delay": 1.5}
{"dest": "MAD", "arr_delay": 20.0}>>>
>{"dest": "GRX", "arr_delay": 2.6}
{"dest": "MAD", "arr_delay": 5.4}
{"dest": "GRX", "arr_delay": 1.5}
{"dest": "MAD", "arr_delay": 20.0}
>{"dest": "GRX", "arr_delay": 2.6}
{"dest": "MAD", "arr_delay": 5.4}
>{"dest": "GRX", "arr_delay": 1.5}
>{"dest": "MAD", "arr_delay": 20.0}
>

```

```

In [8]: agregadosDF = spark.sql("select * from retrasosAgg")

In [9]: columnas = agregadosDF.columns
assert(len(columnas) == 2)
assert("dest" in columnas)
assert("retraso_medio" in columnas)

In [12]: agregadosDF.show() # Ejecuta varias veces esta celda tras enviar el primer mensaje, hasta ver que el DataFrame no es vacío
retraso_medio_GRX_primer_mensaje = 2.6

+-----+
|dest|retraso_medio|
+-----+
| GRX|          2.6|
+-----+

In [15]: # Ejecuta varias veces esta celda tras enviar el cuarto mensaje, hasta ver que el DataFrame ha cambiado
agregadosDF.show()
retraso_medio_GRX_cuarto_mensaje = 2.05
retraso_medio_MAD_cuarto_mensaje = 12.7

+-----+
|dest|retraso_medio|
+-----+
| MAD|          5.4|
| GRX|          2.05|
+-----+

In [13]: # Ejecuta varias veces esta celda tras enviar el segundo mensaje, hasta ver que el DataFrame ha cambiado
agregadosDF.show()
retraso_medio_GRX_segundo_mensaje = 2.6
retraso_medio_MAD_segundo_mensaje = 5.4

+-----+
|dest|retraso_medio|
+-----+
| GRX|          2.6|
+-----+

In [14]: # Ejecuta varias veces esta celda tras enviar el tercer mensaje, hasta ver que el DataFrame ha cambiado
agregadosDF.show()
retraso_medio_GRX_tercer_mensaje = 2.05
retraso_medio_MAD_tercer_mensaje = 5.4

+-----+
|dest|retraso_medio|
+-----+
| MAD|          5.4|
| GRX|          2.6|
+-----+

```