

## Ejercicio Clase 5

1. En el container de Nifi, crear un .sh que permita descargar el archivo yellow\_tripdata\_2021-01.parquet desde

wget -O /home/fpineyro/test/yellow\_tripdata\_2021-01.parquet

[https://data-engineer-edvai-public.s3.amazonaws.com/yellow\\_tripdata\\_2021-01.parquet](https://data-engineer-edvai-public.s3.amazonaws.com/yellow_tripdata_2021-01.parquet) y lo guarde en /home/nifi/ingest. Ejecutarlo.

Script con 'curl' en lugar de wget:

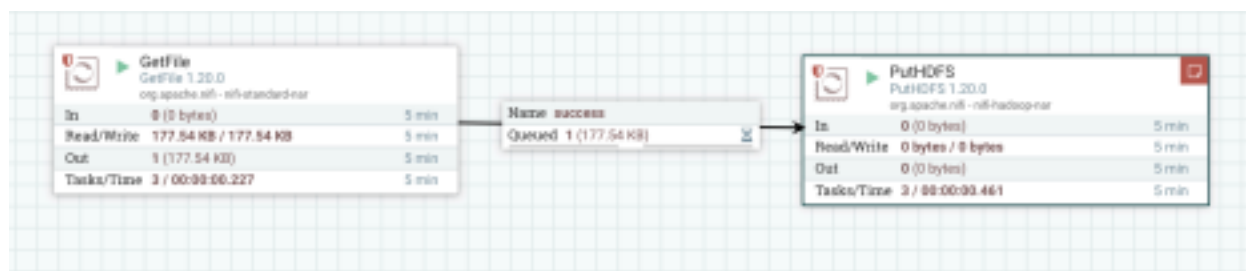
```
nifi@421a768bbd3f:~$ cat landing_parquet.sh
```

```
curl -L -o /home/nifi/ingest/yellow_tripdata_2021-01.parquet
```

```
https://data-engineer-edvai-public.s3.amazonaws.com/yellow\_tripdata\_2021-01.parquet
```



```
nifi@421a768bbd3f:~$ sh landing_parquet.sh
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %             0      0 5828k           0  0:00:03  0:00:03  --:--:-- 5827k
nifi@421a768bbd3f:~$ cd ingest
nifi@421a768bbd3f:~/ingest$ ls -rtl
total 21180
-rw-r--r-- 1 nifi nifi 21686067 Oct  7 09:28 yellow_tripdata_2021-01.parquet
```

2. Por medio de la interfaz gráfica de Nifi, crear un job que tenga dos procesos.
  - a) GetFile para obtener el archivo del punto 1 (/home/nifi/ingest)
  - b) putHDFS para ingestarlo a HDFS (directorio nifi)








Get file:

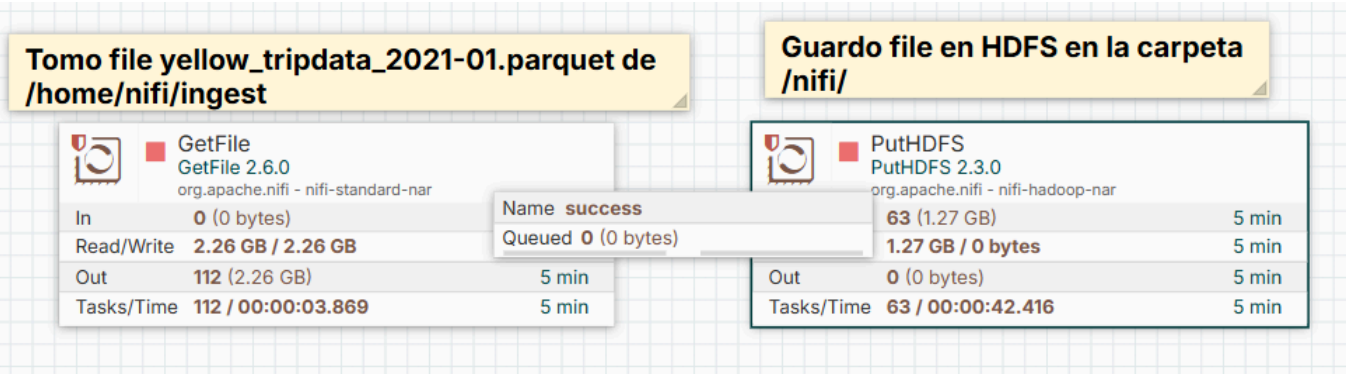
Edit Processor | GetFile 2.6.0

Settings	Scheduling	Properties
Required field		
Property		Value
Input Directory		 /home/nifi/ingest
File Filter		 yellow_tripdata_2021-01.parquet

PutHDFS:

Edit Processor | PutHDFS 2.3.0

Settings	Scheduling	Properties
Required field		
Property		Value
Hadoop Configuration Resources		 /home/nifi/hdfs/core-site.xml, /home...
Kerberos User Service		 No value set
Additional Classpath Resources		 No value set
Directory		 /nifi/
Conflict Resolution Strategy		 replace



Listando el directorio /nifi en HDFS se encuentra el archivo parquet que se descargó en nifi.

```
hadoop@5f1a3da9dadf:/home$ hdfs dfs -ls /nifi
Found 2 items
-rw-r--r-- 1 nifi supergroup 6706 2025-10-07 07:22 /nifi/starwars.avro
-rw-r--r-- 1 nifi supergroup 21686067 2025-10-07 07:37 /nifi/yellow_tripdata_2021-01.parquet
```

3. Con el archivo ya ingestado en HDFS/nifi, escribir las consultas y agregar captura de pantalla del resultado. Para los ejercicios puedes usar SQL mediante la creación de una vista llamada yellow\_tripdata.

También debes chequear el diccionario de datos por cualquier duda que tengas respecto a las columnas del archivo

[https://www.nyc.gov/assets/tlc/downloads/pdf/data\\_dictionary\\_trip\\_records\\_yellow.pdf](https://www.nyc.gov/assets/tlc/downloads/pdf/data_dictionary_trip_records_yellow.pdf)

### 3.1) Mostrar los resultados siguientes

- a. VendorId Integer
- b. Tpep\_pickup\_datetime date
- c. Total\_amount double
- d. Donde el total (total\_amount sea menor a 10 dólares)

```
+-----+-----+-----+
|VendorID|tpep_pickup_datetime|total_amount|
+-----+-----+-----+
|      1|      2020-12-31|         4.3|
|      2|      2020-12-31|         8.3|
|      2|      2020-12-31|        9.96|
|      2|      2020-12-31|         9.3|
|      2|      2020-12-31|         5.8|
|      1|      2020-12-31|         0.0|
|      1|      2020-12-31|         9.3|
|      2|      2020-12-31|         9.8|
|      2|      2020-12-31|         8.8|
|      2|      2020-12-31|        9.96|
+-----+-----+-----+
```

```
SparkSession available as 'spark'.
>>> df = spark.read.parquet("/nifi/yellow_tripdata_2021-01.parquet")
>>> df.show(5)
+-----+-----+-----+-----+
|VendorID|tpep_pickup_datetime|tpep_dropoff_datetime|passenger_count|
+-----+-----+-----+-----+
```

Creo un dataframe con la sentencia: `>>> df_res1 = spark.sql("select cast(VendorID as integer), cast(tppe_pickup_datetime as date), total_amount from yellow_tripdata where total_amount < 10")`

\*El campo `total_amount` no hace falta castearlo, ya era tipo `double`.

```
>>> df_res1 = spark.sql("select cast(VendorID as integer), cast(tppe_pickup_datetime as date), total_amount from yellow_tripdata where total_amount < 10")
```

```
>>> df_res1.printSchema()
root
 |-- VendorID: integer (nullable = true)
 |-- tppe_pickup_datetime: date (nullable = true)
 |-- total_amount: double (nullable = true)
```

```
>>> df_res1.show(10)
+-----+-----+-----+
|VendorID|tppe_pickup_datetime|total_amount|
+-----+-----+-----+
|1|2020-12-31|4.3|
|2|2020-12-31|8.3|
|2|2020-12-31|9.96|
|2|2020-12-31|9.3|
|2|2020-12-31|5.8|
|1|2020-12-31|0.0|
|1|2020-12-31|9.3|
|2|2020-12-31|9.8|
|2|2020-12-31|8.8|
|2|2020-12-31|9.96|
+-----+-----+-----+
only showing top 10 rows
```

3.2)Mostrar los 10 días que más se recaudó dinero (`tppe_pickup_datetime`, `total amount`)

```
+-----+-----+
|tppe_pickup_datetime|sum(total_amount)|
+-----+-----+
|2021-01-28|961322.5600002451|
|2021-01-22|942205.9300002148|
|2021-01-29|937373.5100002222|
|2021-01-21|932444.4500002082|
|2021-01-15|931628.1900002063|
|2021-01-14|926664.0400001821|
|2021-01-27|895259.87000017|
|2021-01-19|890581.4500001629|
|2021-01-07|887670.1600001527|
|2021-01-08|878002.730000146|
+-----+-----+
```

La forma rápida es: >>> df\_3 = spark.sql("select cast(tpep\_pickup\_datetime as date), sum(total\_amount) from yellow\_tripdata group by cast(tpep\_pickup\_datetime as date) order by sum(total\_amount) desc LIMIT 10")

```
>>> df_3 = spark.sql("select cast(tpep_pickup_datetime as date), sum(total_amount) from yellow_tripdata group by cast(tpep_pickup_datetime as date) order by sum(total_amount) desc LIMIT 10")
>>> df_3.show()
+-----+-----+
|tpep_pickup_datetime|sum(total_amount)|
+-----+-----+
|2021-01-28|961322.5600002451|
|2021-01-22|942205.9300002148|
|2021-01-29|937373.5100002222|
|2021-01-21|932444.4500002082|
|2021-01-15|931628.1900002063|
|2021-01-14|926664.0400001821|
|2021-01-27|895259.87000017|
|2021-01-19|890581.4500001629|
|2021-01-07|887670.1600001527|
|2021-01-08|878002.730000146|
+-----+-----+
```

Otra forma es crear un dataframe con las columnas tpep\_pickup\_datetime casteado a date (sino el group by no sale como en la imagen de arriba por los diferentes horarios en el mismo día) y total\_amount que ya es tipo double:

```
>>> df_2 = spark.sql("select cast(tpep_pickup_datetime as date), total_amount from yellow_tripdata")
```

```
>>> df_2 = spark.sql("select cast(tpep_pickup_datetime as date), total_amount from yellow_tripdata")
>>> df_2.show(10)
+-----+-----+
|tpep_pickup_datetime|total_amount|
+-----+-----+
|2020-12-31|11.8|
|2020-12-31|4.3|
|2020-12-31|51.95|
|2020-12-31|36.35|
|2020-12-31|24.36|
|2020-12-31|14.15|
|2020-12-31|17.3|
|2020-12-31|21.8|
|2020-12-31|28.8|
|2020-12-31|18.95|
+-----+-----+
only showing top 10 rows
```

Creo una nueva vista del datagrama df\_2: >>> df\_2.createOrReplaceTempView("yellow\_tripdata\_2")

Creo datagrama df\_3 con el resultado: >>> df\_3 = spark.sql("select tpep\_pickup\_datetime, sum(total\_amount) from yellow\_tripdata\_2 group by tpep\_pickup\_datetime order by sum(total\_amount) desc LIMIT 10")

```
>>> df_3 = spark.sql("select tpep_pickup_datetime, sum(total_amount) from yellow_tripdata_2 group by tpep_pickup_datetime order by sum(total_amount) desc LIMIT 10")
>>> df_3.show()
+-----+-----+
|tpep_pickup_datetime|sum(total_amount)|
+-----+-----+
|2021-01-28|961322.5600002451|
|2021-01-22|942205.9300002148|
|2021-01-29|937373.5100002222|
|2021-01-21|932444.4500002082|
|2021-01-15|931628.1900002063|
|2021-01-14|926664.0400001821|
|2021-01-27|895259.870000017|
|2021-01-19|890581.4500001629|
|2021-01-07|887670.1600001527|
|2021-01-08|878002.730000146|
+-----+-----+
```

3.3) Mostrar los 10 viajes que menos dinero recaudó en viajes mayores a 10 millas (trip\_distance, total\_amount)

```

+-----+-----+
|trip_distance|  total|
+-----+-----+
|          12.68| -252.3|
|          34.35| -176.42|
|          14.75| -152.8|
|          33.96| -127.92|
|           29.1| -119.3|
|          26.94| -111.3|
|          20.08| -107.8|
|          19.55| -102.8|
|          19.16| -90.55|
|          25.83| -88.54|
+-----+-----+

```

Tomo la vista creada en el primer punto yellow\_tripdata. Los campos trip\_distance y total\_amount son de tipo "double". Consulto y guardo en dataframe df\_4:

```

>>> df_4 = spark.sql("select trip_distance, total_amount as total from yellow_tripdata
where trip_distance > 10 order by total_amount asc limit 10")

```

```
>>> df_4 = spark.sql("select trip_distance, total_amount as total from yellow_tripdata
where trip_distance > 10 order by total_amount asc limit 10")
>>> df_4.show(10)
+-----+-----+
|trip_distance|  total|
+-----+-----+
|      12.68| -252.3|
|      34.35| -176.42|
|      14.75| -152.8|
|      33.96| -127.92|
|       29.1| -119.3|
|      26.94| -111.3|
|      20.08| -107.8|
|      19.55| -102.8|
|      19.16|  -90.55|
|      25.83|  -88.54|
+-----+-----+
```

3.4) Mostrar los viajes de más de dos pasajeros que hayan pagado con tarjeta de crédito (mostrar solo las columnas trip\_distance y tpep\_pickup\_datetime)

```
+-----+-----+
|trip_distance|tpep_pickup_datetime|
+-----+-----+
|          2.7|2020-12-31|
|          1.21|2020-12-31|
|          1.16|2020-12-31|
|          0.64|2020-12-31|
|          3.45|2020-12-31|
|          0.52|2020-12-31|
|          1.05|2020-12-31|
|          5.85|2020-12-31|
|          3.7|2020-12-31|
|          4.0|2020-12-31|
+-----+-----+
```



```
>>> df_5 = spark.sql("select trip_distance, cast(tpep_pickup_datetime as date) from  
yellow_tripdata where passenger_count > 2 and payment_type = 1 ")
```

passenger\_count > 2 -> Más de 2 pasajeros  
payment\_type = 1 -> pago en tarjeta de crédito

```
>>> df_5.show(10)
```

trip_distance	tpep_pickup_datetime
6.11	2020-12-31
1.7	2020-12-31
3.15	2020-12-31
10.74	2020-12-31
2.01	2020-12-31
2.85	2020-12-31
1.68	2020-12-31
0.77	2020-12-31
0.4	2020-12-31
16.54	2020-12-31

only showing top 10 rows

3.5) Mostrar los 7 viajes con mayor propina en distancias mayores a 10 millas (mostrar campos tpep\_pickup\_datetime, trip\_distance, passenger\_count, tip\_amount)

trip_distance	tpep_pickup_datetime	passenger_count	tip_amount
427.7	2021-01-20	1	1140.44
267.7	2021-01-03	1	369.4
326.1	2021-01-12	0	192.61
260.5	2021-01-19	1	149.03
11.1	2021-01-31	0	100.0
14.86	2021-01-01	2	99.0
13.0	2021-01-18	0	90.0

```
>>> df_6 = spark.sql("select trip_distance,cast(tpep_pickup_datetime as
date),passenger_count, tip_amount from yellow_tripdata where trip_distance > 10 order by
tip_amount desc limit 7")
```

```
>>> df_6 = spark.sql("select trip_distance,cast(tpep_pickup_datetime as date)
,passenger_count, tip_amount from yellow_tripdata where trip_distance > 10 or
der by tip_amount desc limit 7")
>>> df_6.show(7)
```

trip_distance	tpep_pickup_datetime	passenger_count	tip_amount
427.7	2021-01-20	1.0	1140.44
267.7	2021-01-03	1.0	369.4
326.1	2021-01-12	0.0	192.61
260.5	2021-01-19	1.0	149.03
11.1	2021-01-31	0.0	100.0
14.86	2021-01-01	2.0	99.0
13.0	2021-01-18	0.0	90.0

3.6) Mostrar para cada uno de los valores de RateCodeID, el monto total y el monto promedio. Excluir los viajes en donde RateCodeID es 'Group Ride'

RateCodeID	sum(Total_amount)	avg(Total_amount)
1.0	1.9496468430212937E7	15.606626116946773
4.0	90039.930000000082	74.90842762063296
3.0	67363.260000000043	78.69539719626219
2.0	973635.4700000732	65.52937609369182
99.0	1748.069999999997	48.55749999999999
5.0	255075.08999999086	48.939963545662096

Para que los nombres de columnas coincidan, voy a crear un dataframe.

```
>>> df_8 = spark.sql("select RateCodeID, total_amount as Total_amount from
yellow_tripdata where RateCodeID != 6")
```

Y ahora creo una vista tomando como base `df_8`. >>> df\_8.createOrReplaceTempView("yellow2")

Y finalmente creo y muestro el dataframe nuevo con la suma y promedio.

```
>>> df_9 = spark.sql("select RateCodeID, sum(Total_amount), avg(Total_amount) from yellow2 group
by RateCodeID")
```

```
>>> df_9 = spark.sql("select RateCodeID, sum(Total_amount), avg(Total_amo
nt) from yellow2 group by RateCodeID")
>>> df_9.show()
```

RateCodeID	sum(Total_amount)	avg(Total_amount)
1.0	1.9496468430212937E7	15.606626116946773
4.0	90039.930000000082	74.90842762063296
3.0	67363.260000000043	78.69539719626219
2.0	973635.4700000732	65.52937609369182
99.0	1748.069999999997	48.55749999999999
5.0	255075.08999999086	48.939963545662096