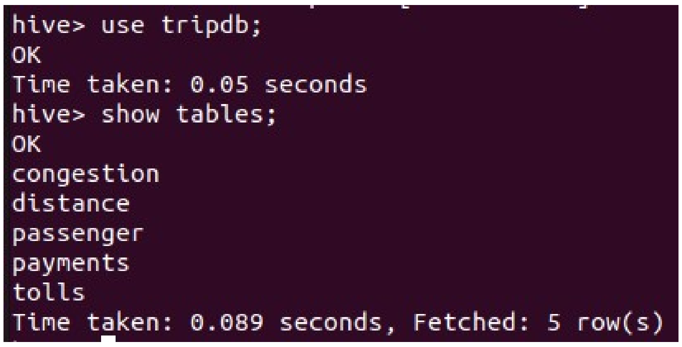
Práctica Hive - Spark

1. Crear las tablas payments, passenger, distance, tolls y congestion en Hive:

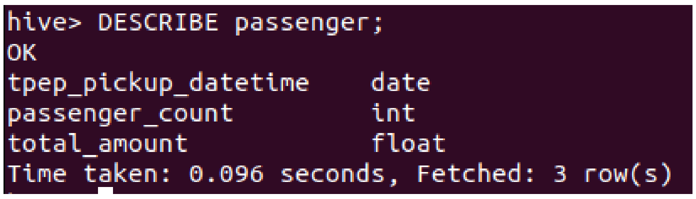
```
CREATE TABLE payments (VendorID int, tpep_pickup_datetime date, payment_type int,
total_amount float)
COMMENT 'Payments'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

CREATE TABLE tolls (tpep_pickup_datetime date, passenger_count int, tolls_amount float,
total_amount float)
COMMENT 'Tolls'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

CREATE TABLE passenger (tpep_pickup_datetime date, passenger_count int, total_amount float)
COMMENT 'Passenger'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

CREATE TABLE congestion (tpep_pickup_datetime date, passenger_count int,
congestion_surcharge float, total_amount float )
COMMENT 'Congestion'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

CREATE TABLE distance (tpep_pickup_datetime date, passenger_count int, trip_distance float,
total_amount float )
COMMENT 'Distance'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
```

Para ver los resultados en la terminal se utiliza *use tripdb* y *show tables.*

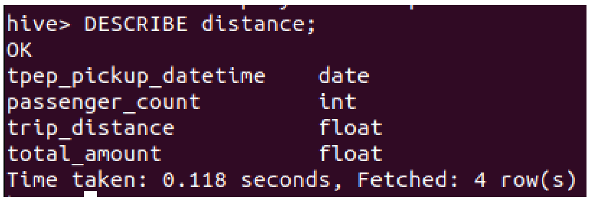


```
hive> use tripdb;
OK
Time taken: 0.05 seconds
hive> show tables;
OK
congestion
distance
passenger
payments
tolls
Time taken: 0.089 seconds, Fetched: 5 row(s)
```

2. En Hive hacer un ‘describe’ de las tablas passengers y distance:

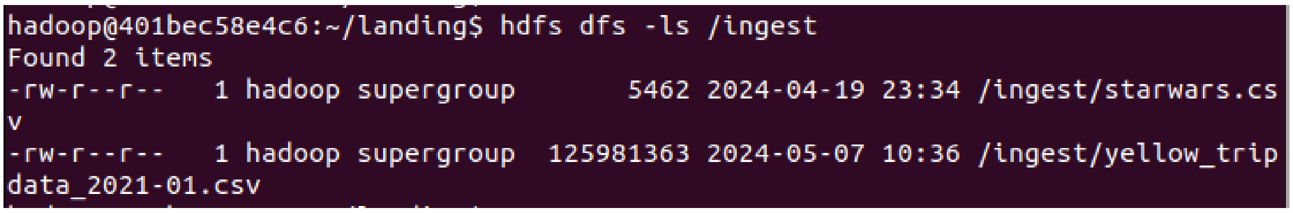


```
hive> DESCRIBE passenger;
OK
tpep_pickup_datetime    date
passenger_count         int
total_amount            float
Time taken: 0.096 seconds, Fetched: 3 row(s)
```



```
hive> DESCRIBE distance;
OK
tpep_pickup_datetime    date
passenger_count         int
trip_distance           float
total_amount            float
Time taken: 0.118 seconds, Fetched: 4 row(s)
```

3. Hacer ingest del file ‘Yellow_tripdata_2021-01.csv’.



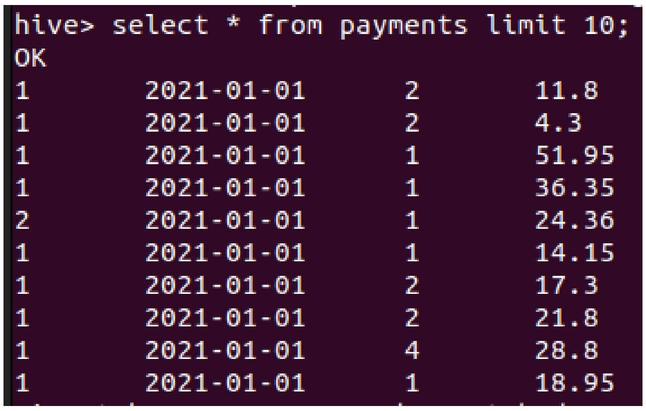
```
hadoop@401bec58e4c6:~/landing$ hdfs dfs -ls /ingest
Found 2 items
-rw-r--r--   1 hadoop supergroup       5462 2024-04-19 23:34 /ingest/starwars.cs
v
-rw-r--r--   1 hadoop supergroup  125981363 2024-05-07 10:36 /ingest/yellow_trip
data_2021-01.csv
```

Para los isguientes ejercicios se debe usar Pyspark. Si se desea practicar más también se puede repetir los mismos en SQL.

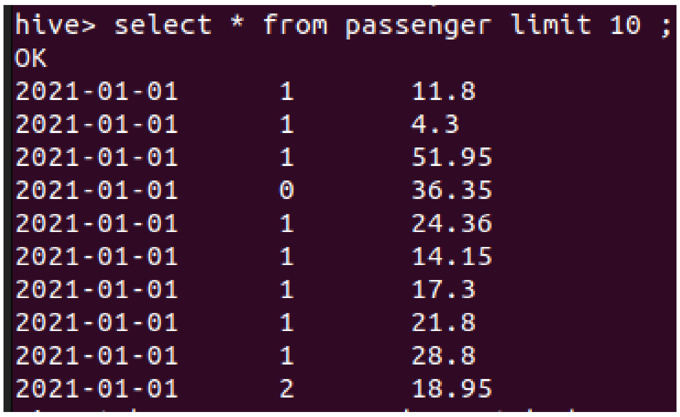
4. Inserción de tablas mediante Pyspark con dataframes:

```
# Carga del csv alojado en hdfs en el dataframe df.
df = spark.read.option("header", "true").csv("/ingest/yellow_tripdata_2021-01.csv")
df.show(1)

# Para la tabla payments:
df_payments = df.select(df.VendorID.cast("int"), df.tpep_pickup_datetime.cast("date"),
df.payment_type.cast("int"), df.total_amount.cast("float"))
df_payments.write.insertInto("tripdb.payments")

# Para la tabla passenger:
df_passenger = df.select(df.tpep_pickup_datetime.cast("date"), df.passenger_count.cast("int"),
df.total_amout.cast("float"))
df_passenger.write.insertInto("tripdb.passenger")

# Para la tabla tolls
df_tolls = df.select(df.tpep_pickup_datetime.cast("date"), df.passenger_count.cast("int"),
df.tolls_amount.cast("float"), df.total_amount.cast("float"))
df_tolls.write.insertInto("tripdb.tolls")

# Para la tabla congestion:
df_congestion = df.select(df.tpep_pickup_datetime.cast("date"), df.passenger_count.cast("int"),
df.congestion_surcharge.cast("float"), df.total_amount.cast("float"))
df_congestion.write.insertInto("tripdb.congestion")

# Para la tabla distance:
df_distance = df.select(df.tpep_pickup_datetime.cast("date"), df.passenger_count.cast("int"),
df.trip_distance.cast("float"), df.total_amount.cast("float"))
df_distance.write.insertInto("tripdb.distance")
```

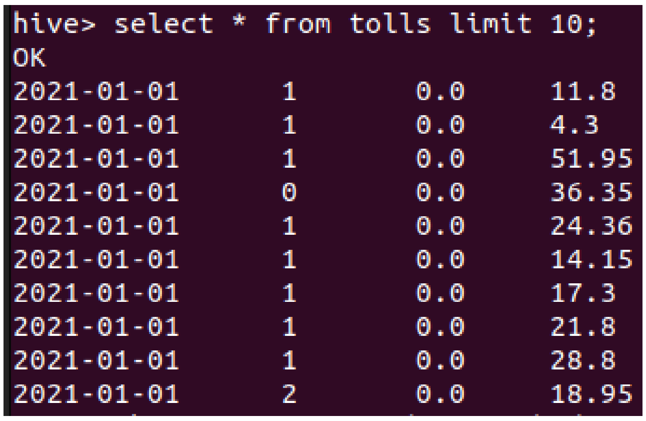
Resultados en Hive:



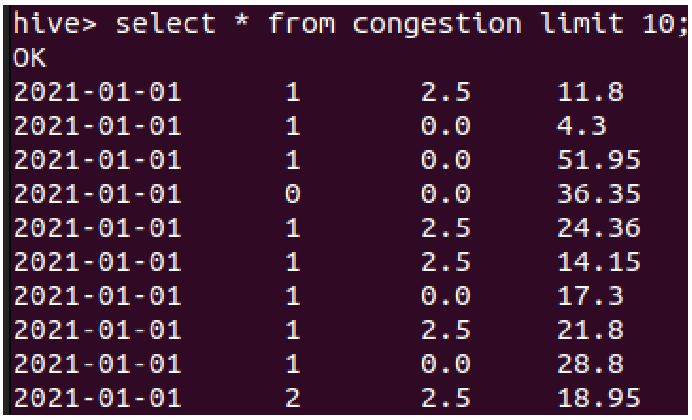
```
hive> select * from payments limit 10;
OK
1       2021-01-01      2       11.8
1       2021-01-01      2       4.3
1       2021-01-01      1       51.95
1       2021-01-01      1       36.35
2       2021-01-01      1       24.36
1       2021-01-01      1       14.15
1       2021-01-01      2       17.3
1       2021-01-01      2       21.8
1       2021-01-01      4       28.8
1       2021-01-01      1       18.95
```



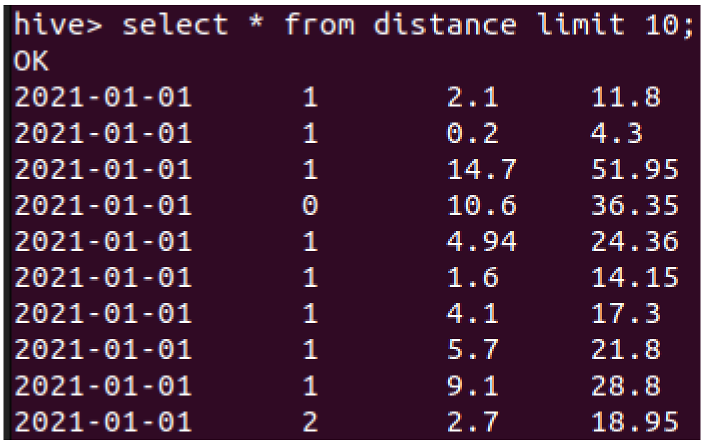
```
hive> select * from passenger limit 10 ;
OK
2021-01-01      1       11.8
2021-01-01      1       4.3
2021-01-01      1       51.95
2021-01-01      0       36.35
2021-01-01      1       24.36
2021-01-01      1       14.15
2021-01-01      1       17.3
2021-01-01      1       21.8
2021-01-01      1       28.8
2021-01-01      2       18.95
```



```
hive> select * from tolls limit 10;
OK
2021-01-01      1       0.0     11.8
2021-01-01      1       0.0     4.3
2021-01-01      1       0.0     51.95
2021-01-01      0       0.0     36.35
2021-01-01      1       0.0     24.36
2021-01-01      1       0.0     14.15
2021-01-01      1       0.0     17.3
2021-01-01      1       0.0     21.8
2021-01-01      1       0.0     28.8
2021-01-01      2       0.0     18.95
```



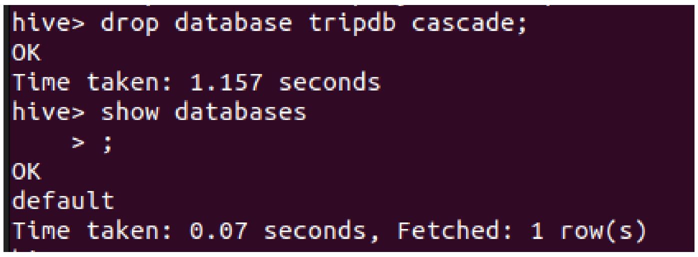
```
hive> select * from congestion limit 10;
OK
2021-01-01      1       2.5     11.8
2021-01-01      1       0.0     4.3
2021-01-01      1       0.0     51.95
2021-01-01      0       0.0     36.35
2021-01-01      1       2.5     24.36
2021-01-01      1       2.5     14.15
2021-01-01      1       0.0     17.3
2021-01-01      1       2.5     21.8
2021-01-01      1       0.0     28.8
2021-01-01      2       2.5     18.95
```



```
hive> select * from distance limit 10;
OK
2021-01-01      1       2.1     11.8
2021-01-01      1       0.2     4.3
2021-01-01      1       14.7    51.95
2021-01-01      0       10.6    36.35
2021-01-01      1       4.94    24.36
2021-01-01      1       1.6     14.15
2021-01-01      1       4.1     17.3
2021-01-01      1       5.7     21.8
2021-01-01      1       9.1     28.8
2021-01-01      2       2.7     18.95
```

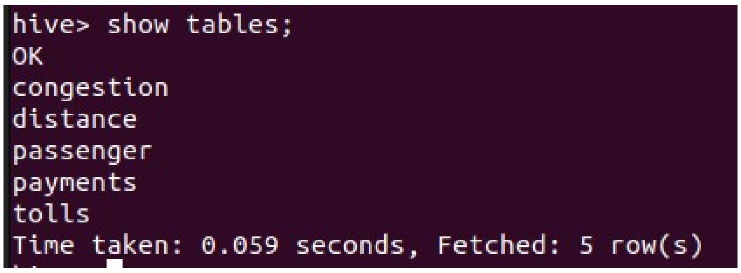
5. Inserción de tablas mediante Pyspark con SQL:

Dropeamos la database



```
hive> drop database tripdb cascade;
OK
Time taken: 1.157 seconds
hive> show databases
    > ;
OK
default
Time taken: 0.07 seconds, Fetched: 1 row(s)
```

Ahora se crea nuevamente con los comandos de Hive que se escribieron al principio.

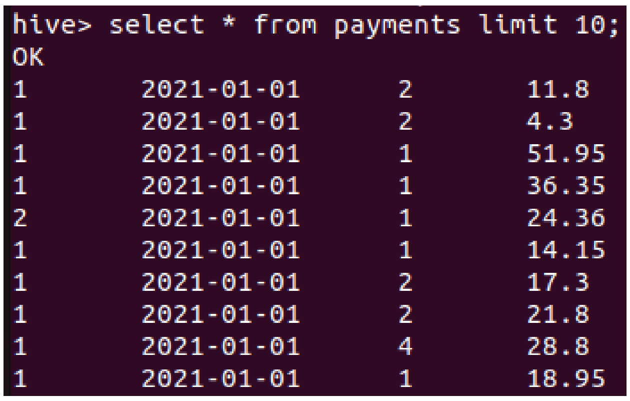


```
hive> show tables;
OK
congestion
distance
passenger
payments
tolls
Time taken: 0.059 seconds, Fetched: 5 row(s)
```

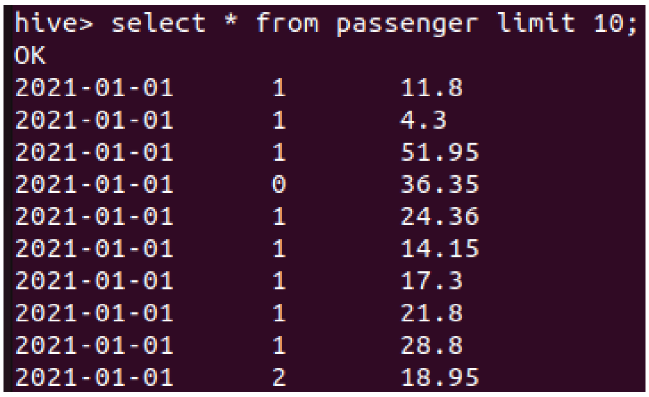
Código para crear las tablas con SQL:

```
# Carga inicial del documento en hdfs a un dataframe llamado df
df = spark.read.option("header", "true").csv("/ingest/yellow_tripdata_2021-01.csv")
df.createOrReplaceTempView("vs_df")

# Carga de la tabla payments
df_payments = spark.sql("select cast(VendorID as int), cast(tpep_pickup_datetime as date),
cast(payment_type as int),  cast(total_amount as float) from vs_df")
df_payments.createOrReplaceTempView("vs_payments_load")
spark.sql("insert into tripdb.payments select * from vs_payments_load")

# Carga de la tabla passenger

df_passengers = spark.sql("select cast(tpep_pickup_datetime as date), cast(passenger_count as int),
cast(total_amount as float) from vs_df")
df_passengers.createOrReplaceTempView("vs_passengers_load")
spark.sql("insert into tripdb.passenger select * from vs_passengers_load")

# Carga de la tabla tolls

df_tolls = spark.sql("select cast(tpep_pickup_datetime as date), cast(passenger_count as int),
cast(tolls_amount as float),  cast(total_amount as float) from vs_df")
df_tolls.createOrReplaceTempView("vs_tolls_load")
spark.sql("insert into tripdb.tolls select * from vs_tolls_load")

# Carga de la tabla congestion

df_congestion = spark.sql("select cast(tpep_pickup_datetime as date), cast(passenger_count as int),
cast(congestion_surcharge as float), cast(total_amount as float) from vs_df")
df_congestion.createOrReplaceTempView("vs_congestion_load")
spark.sql("insert into tripdb.congestion select * from vs_congestion_load")

# Carga de la tabla distance

df_distance = spark.sql("select cast(tpep_pickup_datetime as date), cast(passenger_count as int),
cast(trip_distance as float), cast(total_amount as float) from vs_df")
df_distance.createOrReplaceTempView("vs_distance_load")
spark.sql("insert into tripdb.distance select* from vs_distance_load")
```

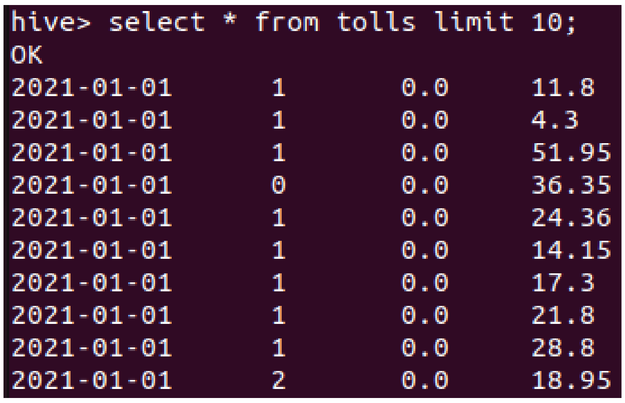
Los resultados en Hive son los mismos que llenando las tablas sin sentencias SQL:



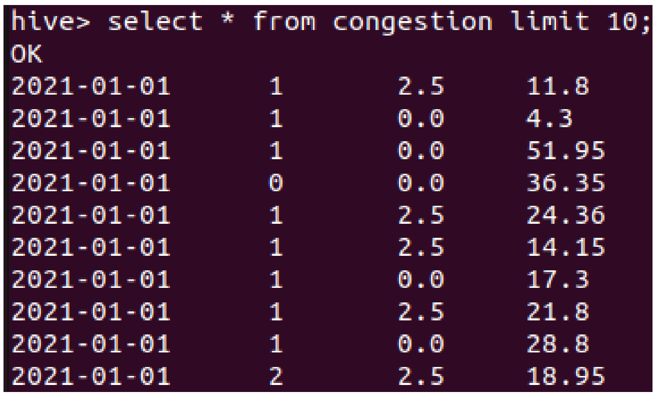
```
hive> select * from payments limit 10;
OK
1       2021-01-01      2       11.8
1       2021-01-01      2       4.3
1       2021-01-01      1       51.95
1       2021-01-01      1       36.35
2       2021-01-01      1       24.36
1       2021-01-01      1       14.15
1       2021-01-01      2       17.3
1       2021-01-01      2       21.8
1       2021-01-01      4       28.8
1       2021-01-01      1       18.95
```



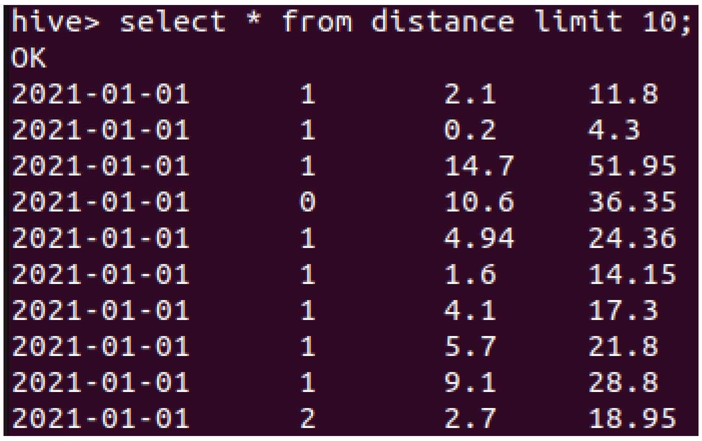
```
hive> select * from passenger limit 10;
OK
2021-01-01      1       11.8
2021-01-01      1       4.3
2021-01-01      1       51.95
2021-01-01      0       36.35
2021-01-01      1       24.36
2021-01-01      1       14.15
2021-01-01      1       17.3
2021-01-01      1       21.8
2021-01-01      1       28.8
2021-01-01      2       18.95
```



```
hive> select * from tolls limit 10;
OK
2021-01-01      1       0.0     11.8
2021-01-01      1       0.0     4.3
2021-01-01      1       0.0     51.95
2021-01-01      0       0.0     36.35
2021-01-01      1       0.0     24.36
2021-01-01      1       0.0     14.15
2021-01-01      1       0.0     17.3
2021-01-01      1       0.0     21.8
2021-01-01      1       0.0     28.8
2021-01-01      2       0.0     18.95
```



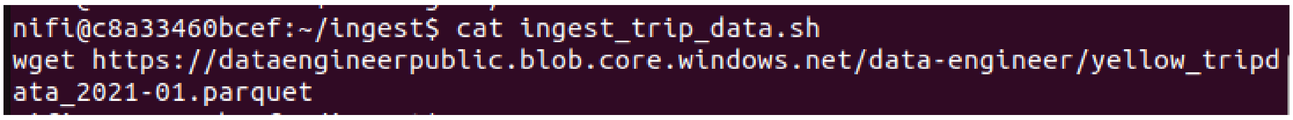
```
hive> select * from congestion limit 10;
OK
2021-01-01      1       2.5     11.8
2021-01-01      1       0.0     4.3
2021-01-01      1       0.0     51.95
2021-01-01      0       0.0     36.35
2021-01-01      1       2.5     24.36
2021-01-01      1       2.5     14.15
2021-01-01      1       0.0     17.3
2021-01-01      1       2.5     21.8
2021-01-01      1       0.0     28.8
2021-01-01      2       2.5     18.95
```



```
hive> select * from distance limit 10;
OK
2021-01-01      1       2.1     11.8
2021-01-01      1       0.2     4.3
2021-01-01      1       14.7    51.95
2021-01-01      0       10.6    36.35
2021-01-01      1       4.94    24.36
2021-01-01      1       1.6     14.15
2021-01-01      1       4.1     17.3
2021-01-01      1       5.7     21.8
2021-01-01      1       9.1     28.8
2021-01-01      2       2.7     18.95
```

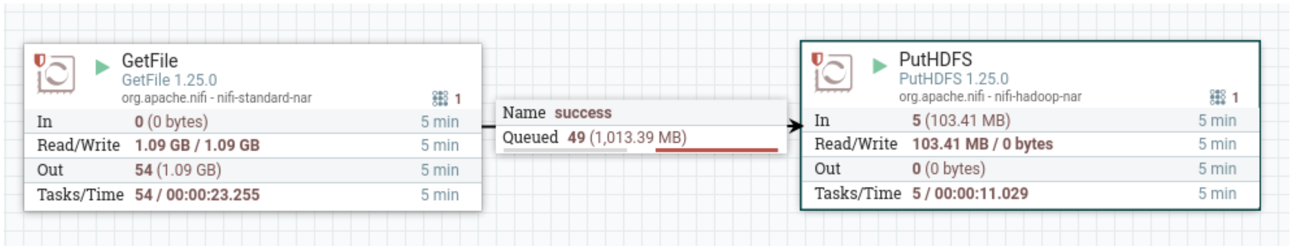
Práctica de transformaciones con Spark y SQL

1. En el container de Nifi crear un .sh que permita descargar el archivo yellow_tripdata_2021-01.parquet.



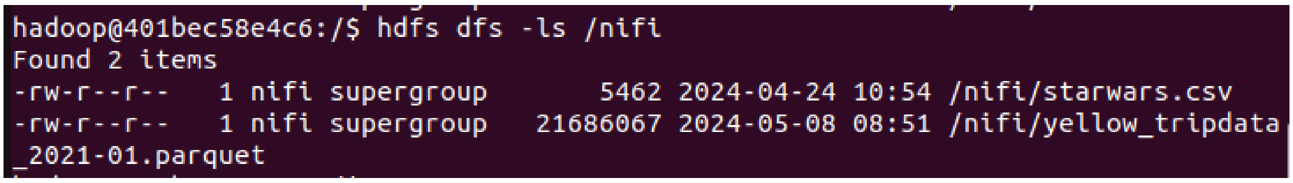
```
nifi@c8a33460bcef:~/ingest$ cat ingest_trip_data.sh
wget https://dataengineerpublic.blob.core.windows.net/data-engineer/yellow_tripd
ata_2021-01.parquet
```

2. Por medio de Nifi crear un job con dos procesos:
a) GetFile para obtener el archivo del punto 1.
b) PutHDFS para ingestarlo a HDFS.



```
GetFile
GetFile 1.25.0
org.apache.nifi - nifi-standard-nar                1
In           0 (0 bytes)                        5 min
Read/Write   1.09 GB / 1.09 GB                  5 min
Out          54 (1.09 GB)                       5 min
Tasks/Time   54 / 00:00:23.255                  5 min

Name  success
Queued  49 (1,013.39 MB)

PutHDFS
PutHDFS 1.25.0
org.apache.nifi - nifi-hadoop-nar                  1
In           5 (103.41 MB)                      5 min
Read/Write   103.41 MB / 0 bytes                5 min
Out          0 (0 bytes)                        5 min
Tasks/Time   5 / 00:00:11.029                   5 min
```

Se puede ver en el contenedor de hadoop hdfs que se cargó el .parquet correctamente.



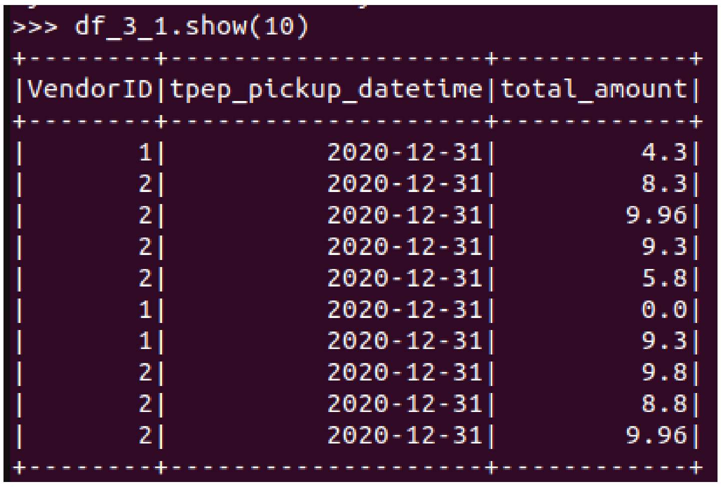
```
hadoop@401bec58e4c6:/$ hdfs dfs -ls /nifi
Found 2 items
-rw-r--r--   1 nifi supergroup       5462 2024-04-24 10:54 /nifi/starwars.csv
-rw-r--r--   1 nifi supergroup   21686067 2024-05-08 08:51 /nifi/yellow_tripdata
_2021-01.parquet
```

3. Con el archivo ingestado en HDFS/nifi, escribir consultas y agregar captura de pantalla del resultado. Para los ejercicios se puede usar SQL mediante la creación de una vista llamada yellow_tripdata.

```
df2 = spark.read.parquet("/nifi/yellow_tripdata_2021-01.parquet")
df2.createOrReplaceTempView("yellow_tripdata")
```

3.1 Mostrar las columnas VendorID int, Tpep_pickup_datetime date, Total_amount double, donde el total_amount < 10 USD.

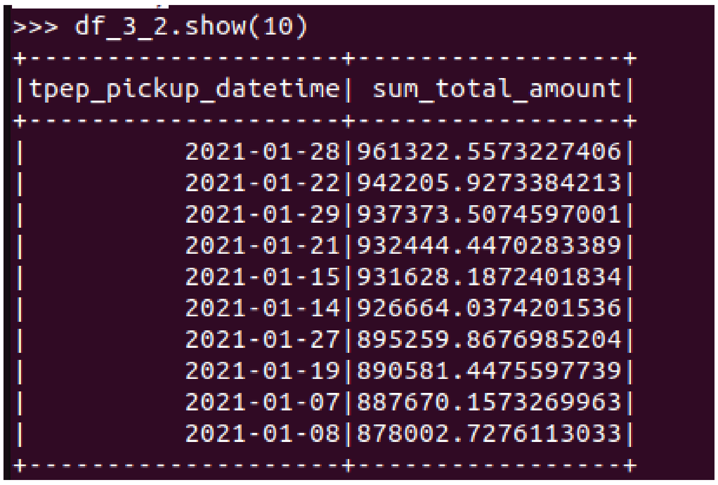
```
df_3_1 = spark.sql("select cast(VendorID as int), cast(tpep_pickup_datetime as date),
cast(total_amount as float) from yellow_tripdata where total_amount <10")
```



```
>>> df_3_1.show(10)
+--------+--------------------+------------+
|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|
+--------+--------------------+------------+
```

3.2 Mostrar los 10 días que más se recaudó dinero (tpep_pickup_datetime, total_amount).

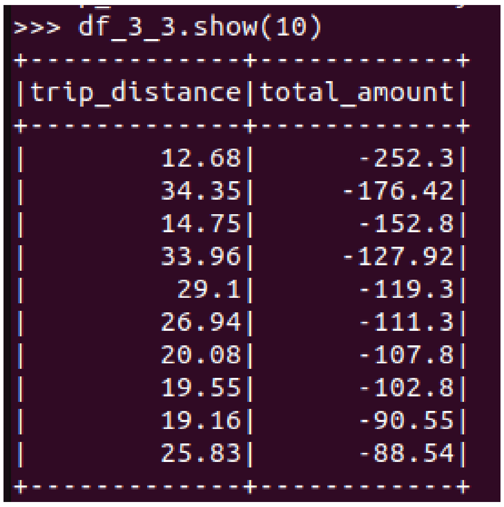
```
df_3_2_cast = spark.sql("select cast(tpep_pickup_datetime as date), cast(total_amount as float)
from yellow_tripdata")
df_3_2_cast.createOrReplaceTempView("3_2_2_cast")
df_3_2 = spark.sql("select tpep_pickup_datetime, sum(total_amount) as sum_total_amount  from
3_2_2_cast group by tpep_pickup_datetime order by sum_total_amount desc")
```



```
>>> df_3_2.show(10)
+--------------------+-----------------+
|tpep_pickup_datetime| sum_total_amount|
+--------------------+-----------------+
|          2021-01-28|961322.5573227406|
|          2021-01-22|942205.9273384213|
|          2021-01-29|937373.5074597001|
|          2021-01-21|932444.4470283389|
|          2021-01-15|931628.1872401834|
|          2021-01-14|926664.0374201536|
|          2021-01-27|895259.8676985204|
|          2021-01-19|890581.4475597739|
|          2021-01-07|887670.1573269963|
|          2021-01-08|878002.7276113033|
+--------------------+-----------------+
```

3.3 Mostrar los 10 viajes que menos dinero recaudó en viajes mayores a 10 millas (trip_distance, total amount)

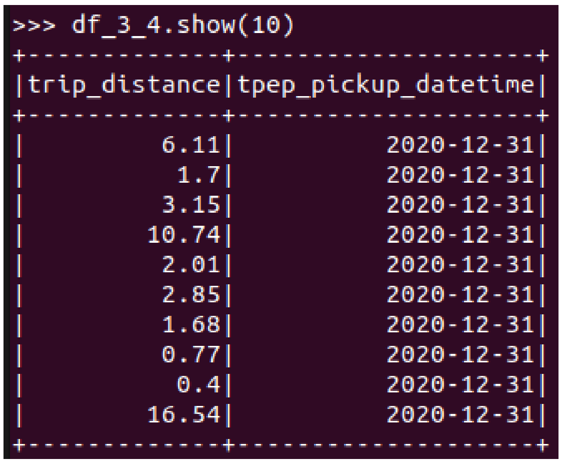
```
df_3_3_cast = spark.sql("select cast(trip_distance as float), cast(total_amount as float) from
yellow_tripdata")
df_3_3_cast.createOrReplaceTempView("3_2_3_cast")
df_3_3 = spark.sql("select trip_distance, total_amount from 3_2_3_cast where trip_distance > 10
order by total_amount asc")
```



```
>>> df_3_3.show(10)
+-------------+------------+
|trip_distance|total_amount|
+-------------+------------+
|        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 pajeros que hayan pagado con tarjeta de crédito. (trip_distance, tpep_pickup_datetime)

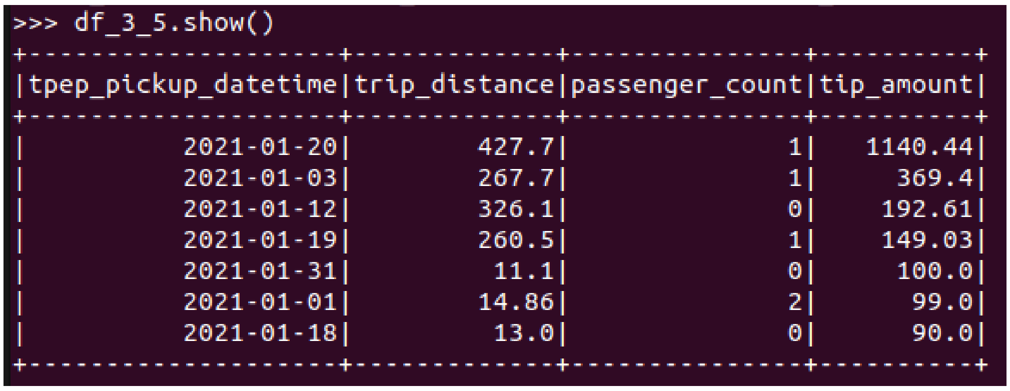
```
df_3_4_cast = spark.sql("select cast(trip_distance as float), cast(tpep_pickup_datetime as date),
cast(passenger_count as int), cast(payment_type as int) from yellow_tripdata")
df_3_4_cast.createOrReplaceTempView("3_2_4_cast")
df_3_4 = spark.sql("select trip_distance, tpep_pickup_datetime from 3_2_4_cast where
passenger_count > 2 and payment_type = 1")
```



```
>>> df_3_4.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|
+-------------+--------------------+
```

3.5 Mostrar los 7 viajes con mayor propina en distancias mayores a 10 millas (tpep_pickup_datetime, trip_distance, passenger_count, tip_amount)

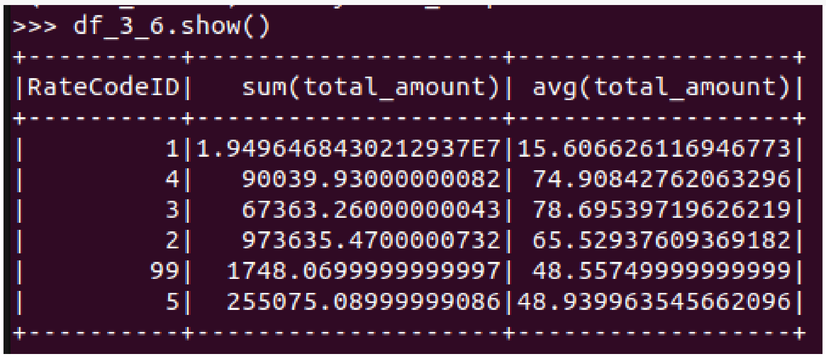
```
df_3_5 = spark.sql("select cast(tpep_pickup_datetime as date), cast(trip_distance as float),
cast(passenger_count as int), cast (tip_amount as float) from yellow_tripdata where trip_distance >
10 order by tip_amount DESC LIMIT 7")
```



```
>>> df_3_5.show()
+--------------------+-------------+---------------+----------+
|tpep_pickup_datetime|trip_distance|passenger_count|tip_amount|
+--------------------+-------------+---------------+----------+
|          2021-01-20|        427.7|              1|   1140.44|
|          2021-01-03|        267.7|              1|     369.4|
|          2021-01-12|        326.1|              0|    192.61|
|          2021-01-19|        260.5|              1|    149.03|
|          2021-01-31|         11.1|              0|     100.0|
|          2021-01-01|        14.86|              2|      99.0|
|          2021-01-18|         13.0|              0|      90.0|
+--------------------+-------------+---------------+----------+
```

3.6 Mostrar para cada uno de los valores RateCodeID, el monto total y el monto promedio. Excluir los viajes donde RateCodeID es ‘Group ride’

```
df_3_6 = spark.sql(" SELECT  CAST(RateCodeID as int),  SUM(total_amount),
AVG(total_amount) FROM yellow_tripdata WHERE RateCodeID != 6 GROUP BY RateCodeID")
```



```
>>> df_3_6.show()
+----------+-------------------+------------------+
|RateCodeID|  sum(total_amount)| avg(total_amount)|
+----------+-------------------+------------------+
|         1|1.9496468430212937E7|15.606626116946773|
|         4|   90039.93000000082| 74.90842762063296|
|         3|   67363.26000000043| 78.69539719626219|
|         2|   973635.4700000732| 65.52937609369182|
|        99|  1748.0699999999997| 48.55749999999999|
|         5|  255075.08999999086|48.939963545662096|
+----------+-------------------+------------------+
```