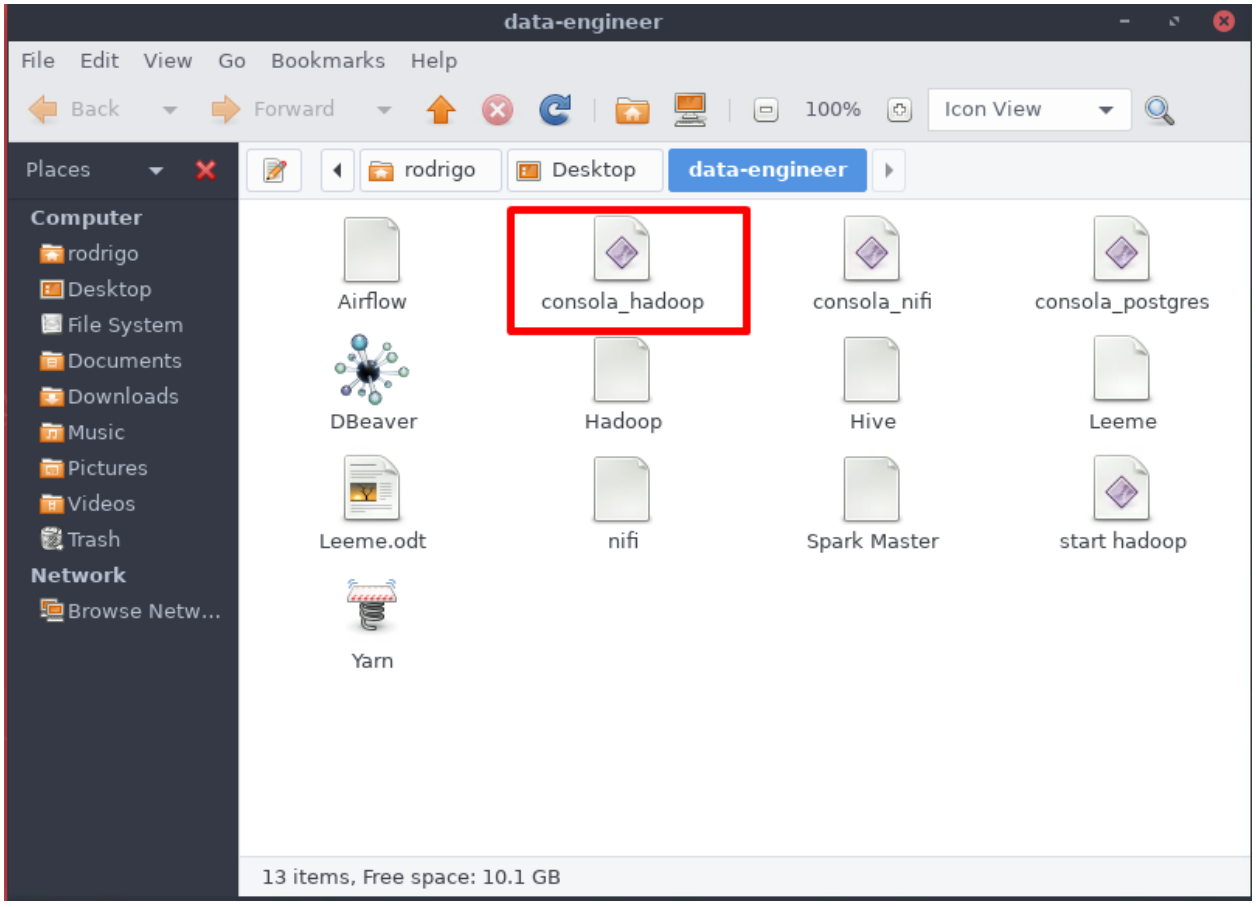
Sqoop

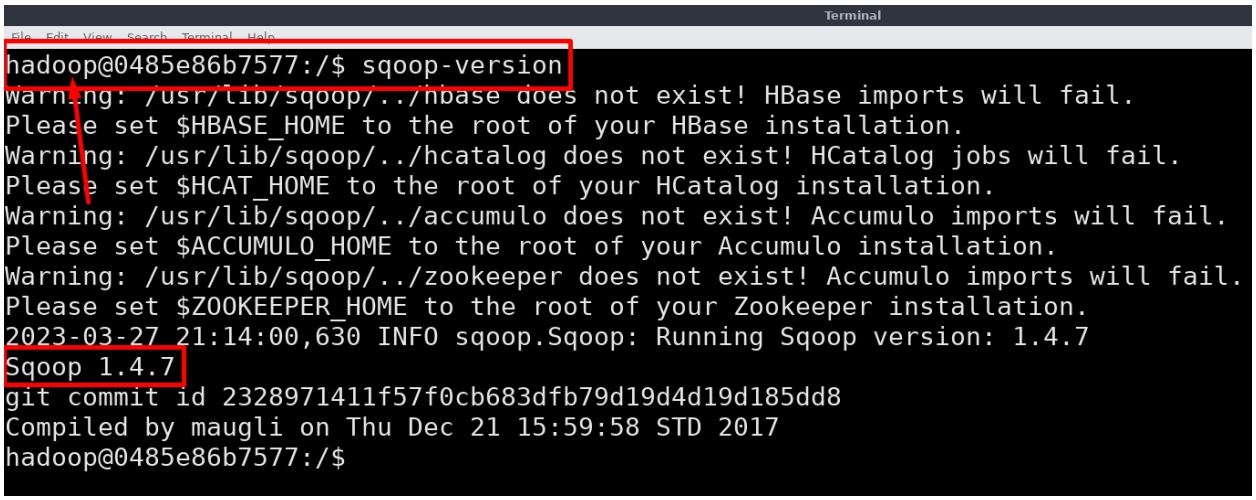
NOTA: por favor antes de continuar revisar la dirección IP y el puerto de Postgres, lo pueden hacer con el siguiente comando:

Docker inspect edvai_postgres

Para comenzar a utilizar sqoop debemos ingresar a la consola de Hadoop:



Verificar que estemos logueados con el usr hadoop y ya podemos comenzar a probar los siguientes comandos:



Verificar funcionamiento y versión:

- **Sqoop-version**

Listar databases:

**sqoop list-databases **
**—connect jdbc:postgresql://172.17.0.3:5432/northwind **
—username postgres -P

Listar tablas:

**sqoop list-tables **
**—connect jdbc:postgresql://172.17.0.3:5432/northwind **
—username postgres -P

Ejecutar Queries:

```
sqoop eval \
—connect jdbc:postgresql://172.17.0.3:5432/northwind \
—username postgres \
—P \
—query "select * from region limit 10"
```

Importar tablas:

```
sqoop import \
  —connect jdbc:postgresql://172.17.0.3:5432/northwind \
  —username postgres\
  —table region\
  —m 1 \
  —P \
  —target-dir /sqoop/ingest \
  —as-parquetfile \
  —delete-target-dir
```

Importar tablas con filtro:

```
sqoop import \
  —connect jdbc:postgresql://172.17.0.3:5432/northwind \
  —username postgres\
  —table region\
  —m 1 \
  —P \
  —target-dir /sqoop/ingest/southern \
  —as-parquetfile \
  —where "region_description = 'Southern' " \
  —delete-target-dir
```

Importar tablas desde una query:

```
sqoop import \
  —connect jdbc:postgresql://172.17.0.3:5432/northwind \
  —username postgres\
  —query "select * from region where region_id = 3 AND \$CONDITIONS"\
  —m 1 \
  —P \
  —target-dir /sqoop/ingest \
  —as-parquetfile \
  —delete-target-dir
```

Una vez que hayamos hecho el import nos quedará un archivo parquet en la siguiente ruta: **/sqoop/ingest**

Por lo que luego ya podemos ingresar a spark para comenzar a crear un dataframe en base a esa data.

Creamos un dataframe leyendo el parquet que acabamos de importar con sqoop
df = spark.read.parquet("/sqoop/ingest/*.parquet")

Luego podemos revisar el esquema
df.printSchema()

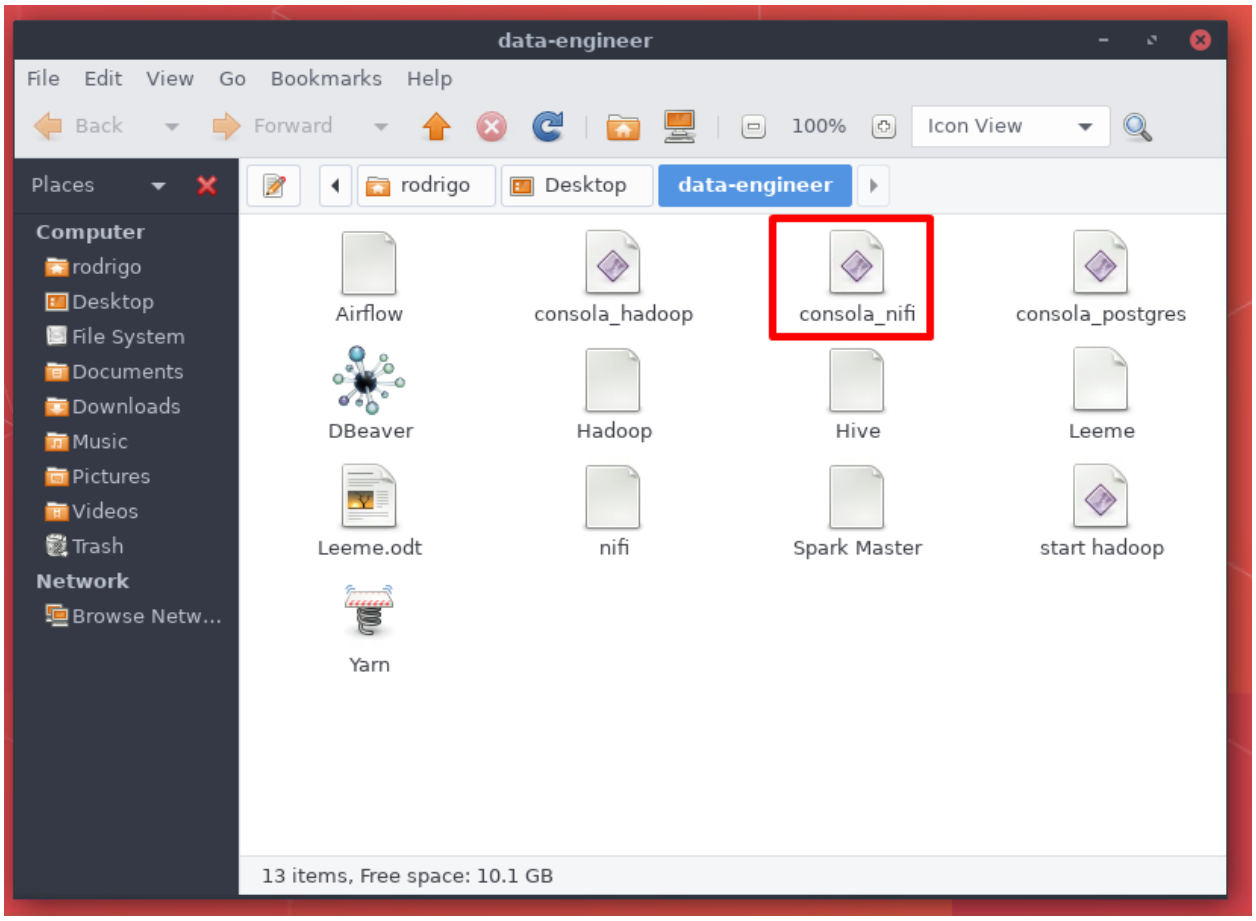
Podemos revisar si se encuentra datos dentro del dataframe:

```
>>> df.show(5)
+————+———————+———————+—————+————+
|order_id|customer_id|ship_country|unit_price|quantity|
+————+———————+———————+—————+————+
|  10248|    VINET|    France|     14.0|     12|
|  10248|    VINET|    France|      9.8|     10|
|  10248|    VINET|    France|     34.8|      5|
|  10249|    TOMSP|   Germany|     18.6|      9|
|  10249|    TOMSP|   Germany|     42.4|     40|
+————+———————+———————+—————+————+
only showing top 5 rows
```

APACHE nifi

Para comenzar debemos ingresar a la consola nifi o ejecutar desde la ventana de terminal de la vm el siguiente comando:

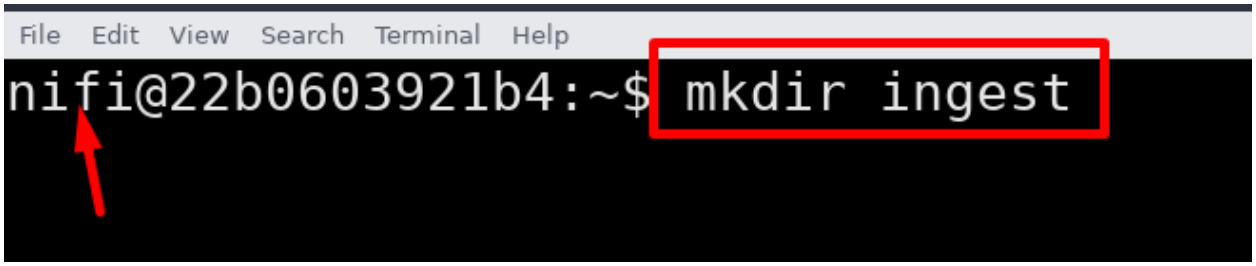
Docker exec -it nifi bash



Una vez que ingresamos ir a la carpeta /home/nifi

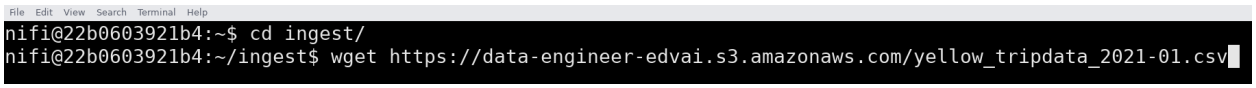
Y creamos 3 carpetas:

- Ingest
- Bucket
- hadoop



Ingresamos a la carpeta ingest y bajamos el siguiente archivo con el siguiente comando:

wget https://data-engineer-edvai.s3.amazonaws.com/yellow_tripdata_2021-01.csv



Luego vamos a la carpeta hadoop y creamos los siguientes archivos:

- core-site.xml
- hdfs-site.xml

De la sigueitne manera:

Cat > core-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->


<configuration>
        <property>
                <name>fs.defaultFS</name>
```

```
            <value>hdfs://172.17.0.2:9000</value>
        </property>
</configuration>
```

Y luego Ctrl + D para finalizar la creación del archivo

```
Cat > hdfs-site.xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->


<configuration>

        <property>
                <name>dfs.replication</name>
                <value>1</value>
        </property>

        <property>
                <name>dfs.name.dir</name>
                <value>file:///home/hadoop/hadoopdata/hdfs/namenode</value>
        </property>

        <property>
                <name>dfs.data.dir</name>
                <value>file:///home/hadoop/hadoopdata/hdfs/datanode</value>
        </property>
</configuration>
```

Y luego Ctrl + D para finalizar la creación del archivo

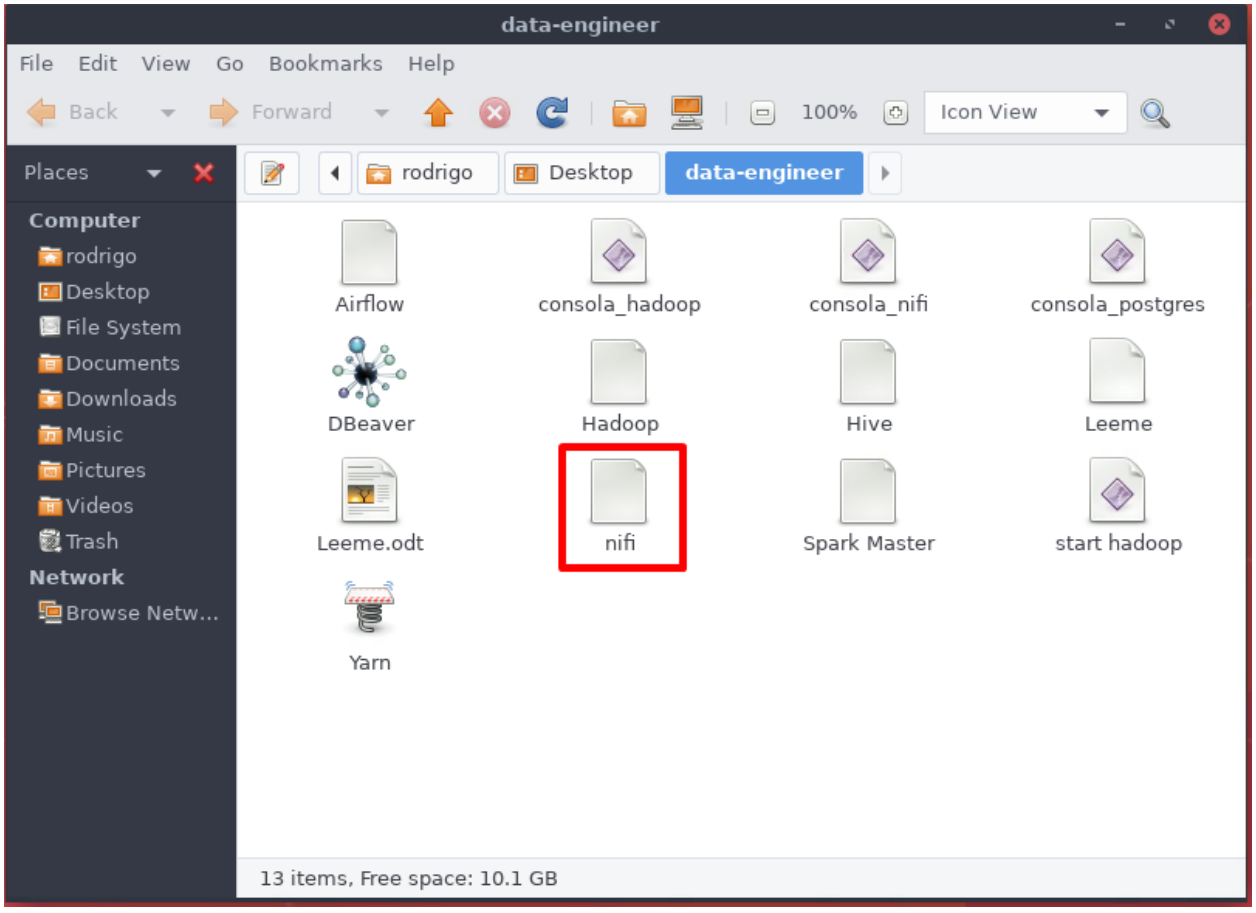
Luego ingresamos a la UI de nifi desde el ícono nifi o ingresamos a firefox e ingresamos la siguiente dirección:

https://localhost:8443/nifi/

El usr y la pass son las siguientes:
Generated Username [d30eb1a2-3bfe-4c85-9ea4-9562915a70e6]
Generated Password [NvxFSKesWIiU1K4XL1AQJwovv9z7TW4h]

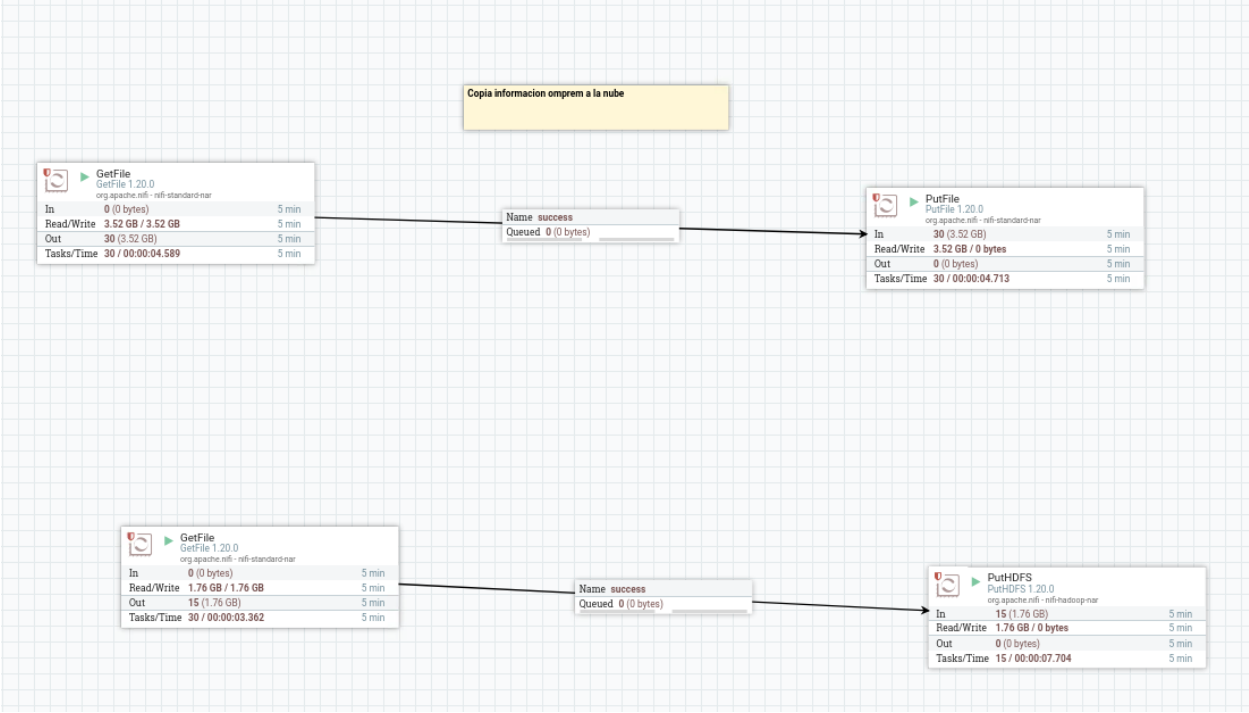
En caso de no funcionar ir a la terminal de la vm y ejecutar el siguiente comando

docker logs nifi | grep Generated



Una vez que ingresamos vamos a generar 4 procesadores:

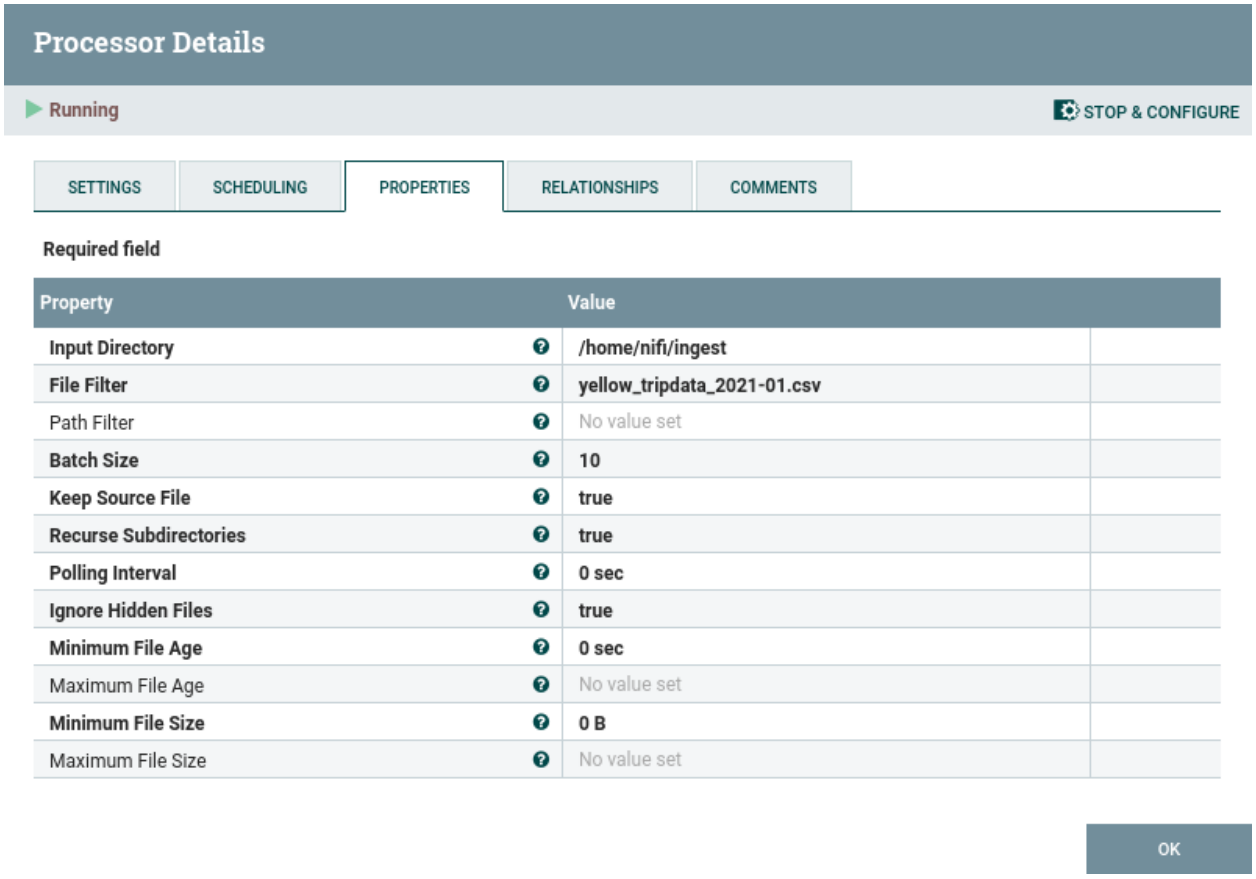
- 2 GetFile
- 1 Putfile
- 1 PutHDFS



El primer proceso va a tomar datos desde /ingest y lo va a copiar a /bucket de la siguiente manera:

GETFILE

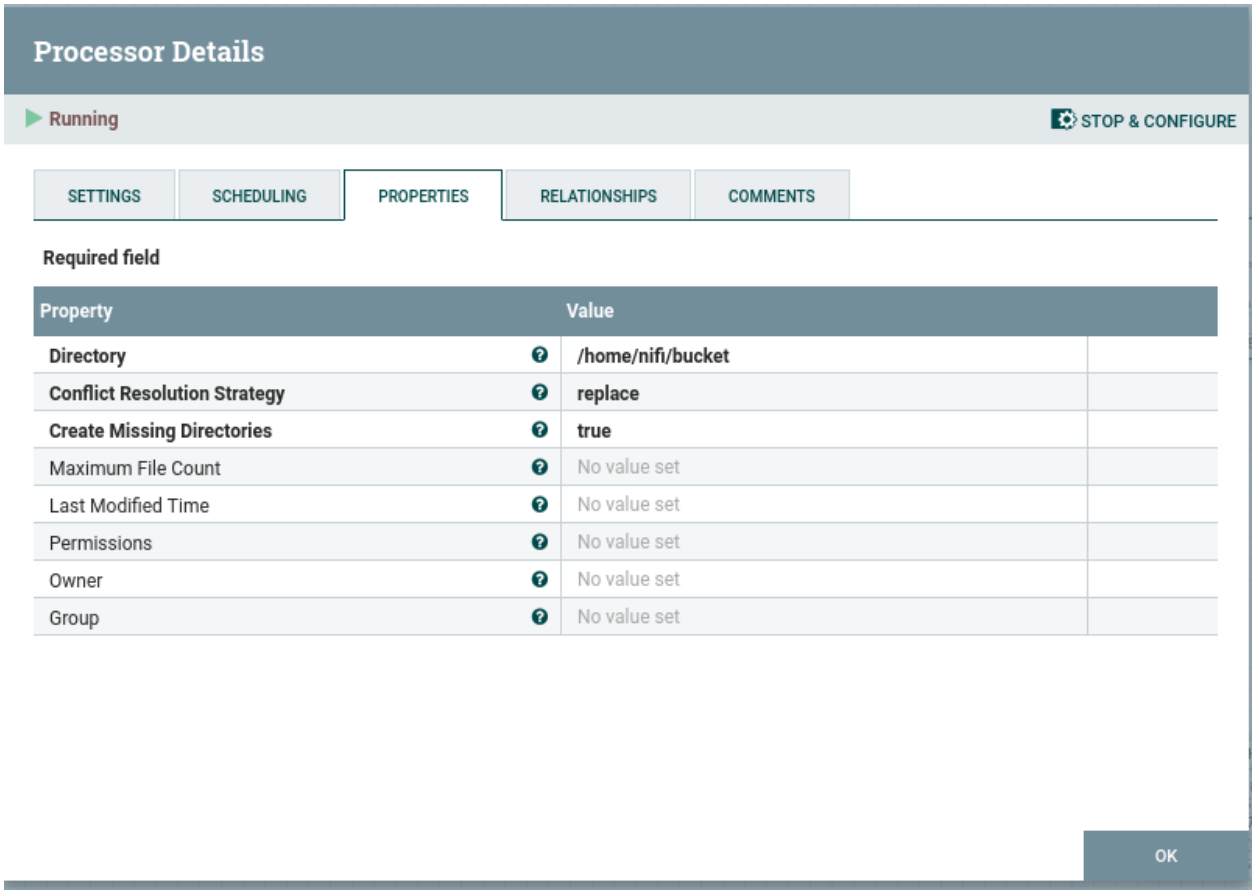
input directory: /home/nifi/ingest
File filter: yellow_tripdata_2021-01.csv (o dejar por default que no pone ningún filtro)
Scheduling: 10 seg



PutFile

Directory: home/nifi/bucket
Relationships:

- Failure: retry and terminate
- Success: terminate



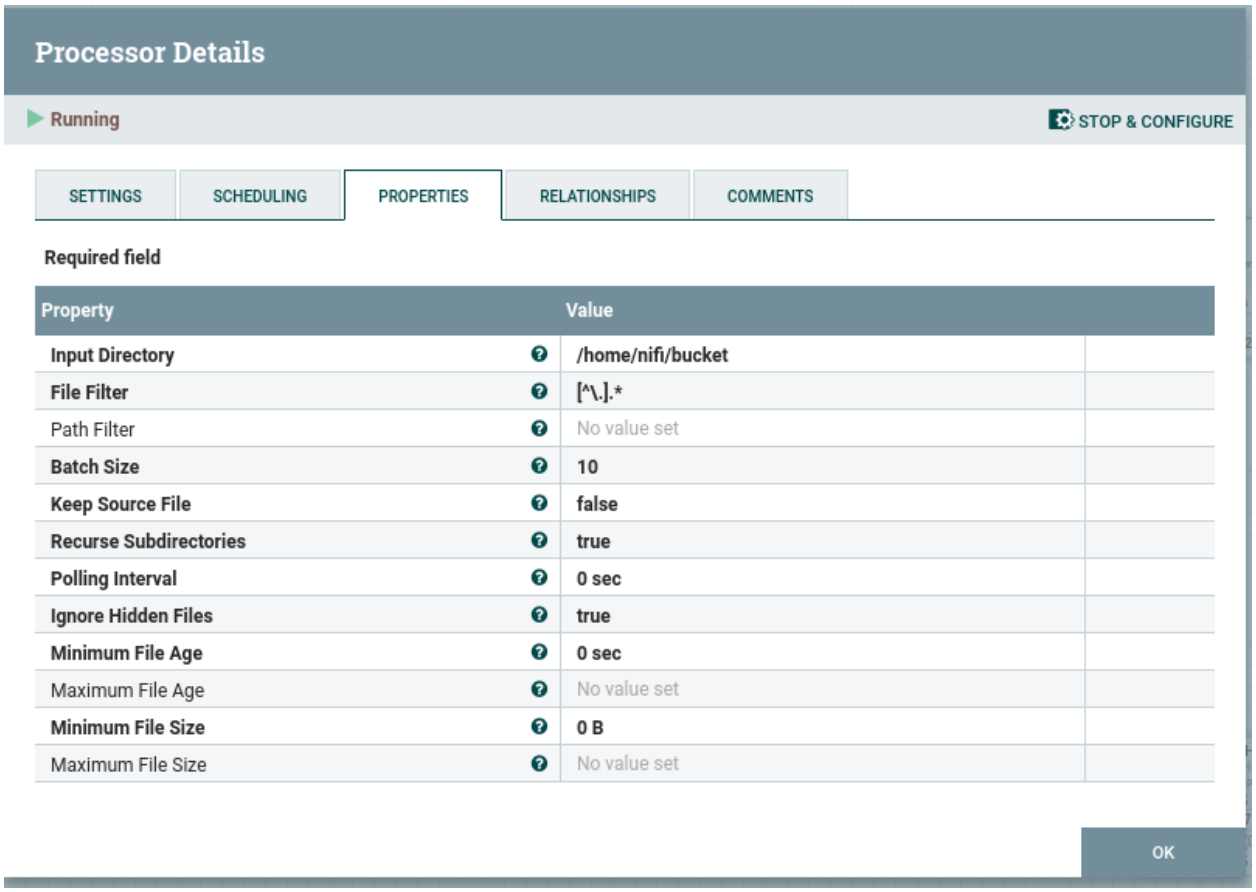
Lo linkeamos con el put file a través de la flecha

Luego ya lo podemos hacer correr, para verificar que funcionó OK debemos revisar la carpeta /home/nifi/bucket y ver si se encuentra el archivo yellow_tripdata_2021-01.csv

Ahora nos queda copiar el archivo /home/nifi/bucket/yellow_tripdata_2021-01.csv a la carpeta /nifi de HDFS

Getfile

Input directory: /home/nifi/bucket
File filter: dejarlo por default para que no filtre ningún archivo
Scheduling: 10 seg

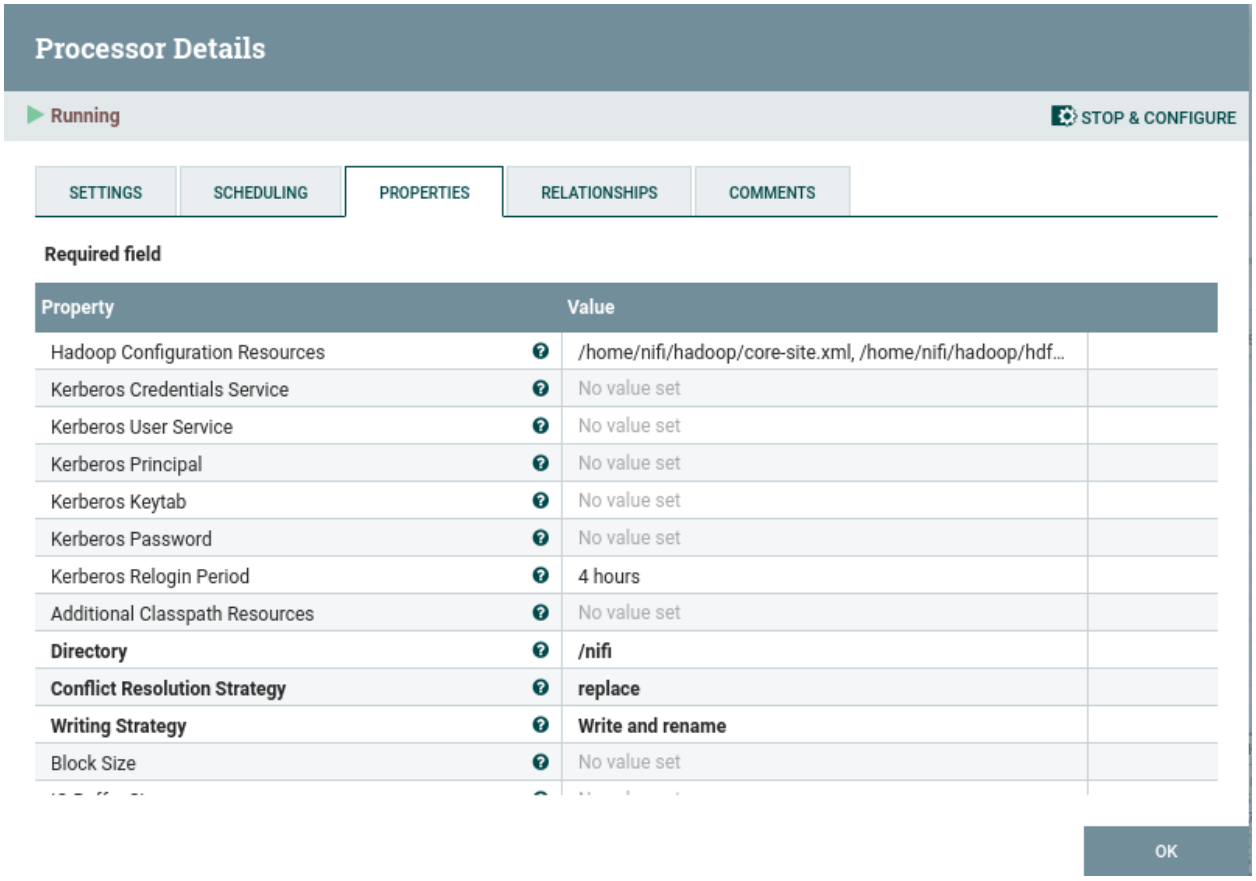


PutHDFS
Luego creamos el procesador PutHDFS de la siguiente manera:

Hadoop configuration resources (archivos de configuración de hadoop):
/home/nifi/hadoop/core-site.xml, /home/nifi/hadoop/hdfs-site.xml

Directory: /nifi
Conflict resolution strategy: replace
Relationships:

- Failure: retry and terminate
- Success: terminate



Luego tenemos que revisar que nifi pueda escribir en el directorio /nifi
Para esto ingresamos en la consola hadoop

Y escribimos en la consola hdfs dfs -ls /

Ahí revisamos que otros usuarios no pueden escribir en esa carpeta, por lo que vamos a modificar los permisos

```
hadoop@0485e86b7577:/$ hdfs dfs -ls /
Found 7 items
drwxr-xr-x   - hadoop supergroup          0 2023-03-21 09:02 /ingest
drwxr-xr-x   - hadoop supergroup          0 2022-04-26 19:51 /inputs
drwxr-xr-x   - hadoop supergroup          0 2022-01-22 21:35 /logs
drwxr-xr-x   - hadoop supergroup          0 2023-03-27 20:47 /nifi
drwxr-xr-x   - hadoop supergroup          0 2023-03-27 19:47 /sqoop
drwxrwxr-x   - hadoop supergroup          0 2022-05-02 20:46 /tmp
drwxr-xr-x   - hadoop supergroup          0 2022-01-23 13:15 /user
```

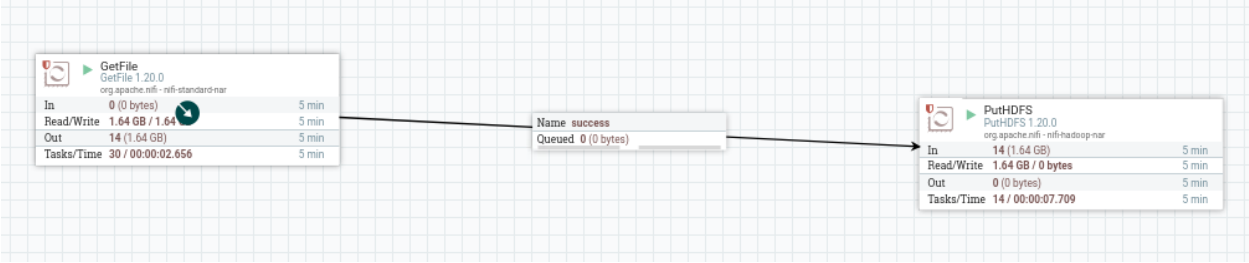
Modificamos los permisos con el siguiente comando:

Hdfs dfs -chmod 777 /nifi

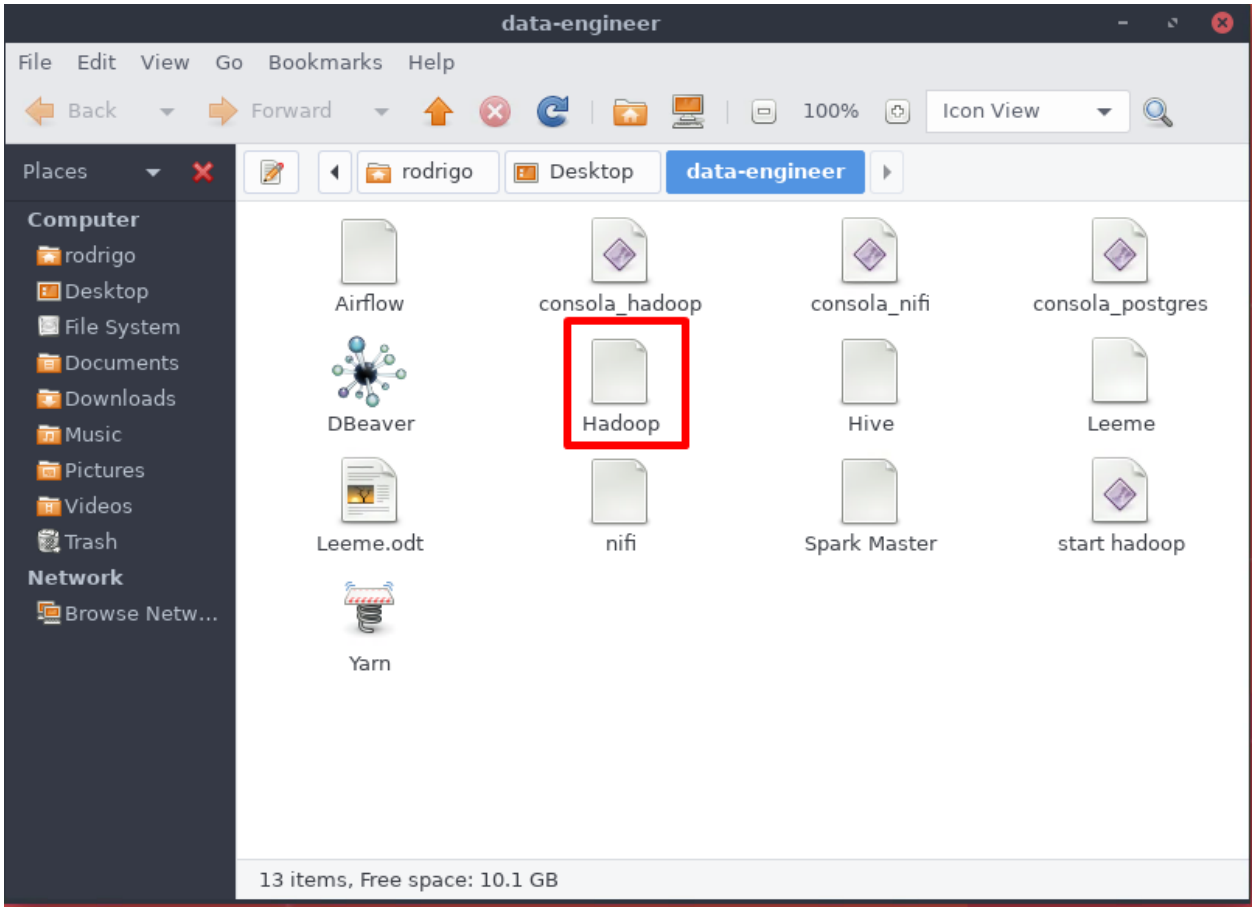
Luego verificamos que afectivamente el directorio /nifi tengan permisos para escribir otros usuarios

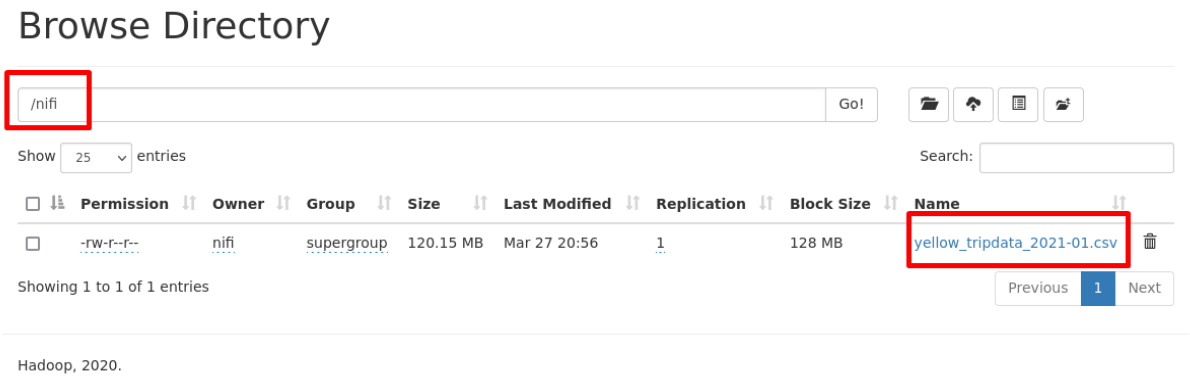
```
hadoop@0485e86b7577:/$ hdfs dfs -chmod 777 /nifi
hadoop@0485e86b7577:/$ hdfs dfs -ls /
Found 7 items
drwxr-xr-x   - hadoop supergroup          0 2023-03-21 09:02 /ingest
drwxr-xr-x   - hadoop supergroup          0 2022-04-26 19:51 /inputs
drwxr-xr-x   - hadoop supergroup          0 2022-01-22 21:35 /logs
drwxrwxrwx   - hadoop supergroup          0 2023-03-27 20:47 /nifi
drwxr-xr-x   - hadoop supergroup          0 2023-03-27 19:47 /sqoop
drwxrwxr-x   - hadoop supergroup          0 2022-05-02 20:46 /tmp
drwxr-xr-x   - hadoop supergroup          0 2022-01-23 13:15 /user
```

Ahora si podemos darle start a la segunda parte del proces que me va a permitir copiar el archivo yellow_tripdata_2021-01.csv desde la carpeta /nifi/bucket al directorio /nifi



Para verificar que efectivamente copio el archivo lo podemos hacer desde la UI hadoop





O desde la consola hadoop con el siguiente comando:

```
hadoop@0485e86b7577:/$ hdfs dfs -ls /nifi
Found 1 items
-rw-r--r--   1 nifi supergroup  125981363 2023-03-27 21:50 /nifi/yellow_tripdata_2021-01.csv
hadoop@0485e86b7577:/$
```

Por lo que luego ya podemos ingresar a spark para comenzar a crear un dataframe en base a esa data.

Creamos un dataframe leyendo el parquet que acabamos de importar con sqoop
df = spark.read.option("header", "true").csv("/nifi/*.csv")

Luego podemos revisar el esquema
>>> df.printSchema()
root
|– VendorID: string (nullable = true)
|– tpep_pickup_datetime: string (nullable = true)
|– tpep_dropoff_datetime: string (nullable = true)
|– passenger_count: string (nullable = true)
|– trip_distance: string (nullable = true)
|– RatecodeID: string (nullable = true)
|– store_and_fwd_flag: string (nullable = true)
|– PULocationID: string (nullable = true)
|– DOLocationID: string (nullable = true)
|– payment_type: string (nullable = true)
|– fare_amount: string (nullable = true)
|– extra: string (nullable = true)
|– mta_tax: string (nullable = true)
|– tip_amount: string (nullable = true)
|– tolls_amount: string (nullable = true)
|– improvement_surcharge: string (nullable = true)
|– total_amount: string (nullable = true)
|– congestion_surcharge: string (nullable = true)

Podemos revisar si se encuentra datos dentro del dataframe:
>> df.show(3)

```
+--------+--------------------+---------------------+---------------+-------------+----------+------------------+---------
----+------------+------------+-----------+-----+-------+----------+------------+---------------------+------------+----
---------------+
|VendorID|tpep_pickup_datetime|tpep_dropoff_datetime|passenger_count|trip_distance|
RatecodeID|store_and_fwd_flag|PULocationID|DOLocationID|payment_type|fare_amount|
extra|mta_tax|tip_amount|tolls_amount|improvement_surcharge|total_amount|congestio
n_surcharge|
+--------+--------------------+---------------------+---------------+-------------+----------+------------------+---------
----+------------+------------+-----------+-----+-------+----------+------------+---------------------+------------+----
---------------+
```

```
|       1| 2021-01-01 00:30:10|  2021-01-01 00:36:12|            1|        2.10|        1|                N|
142|          43|           2|          8|    3|    0.5|         0|          0|                0.3|       11.8|
2.5|
|       1| 2021-01-01 00:51:20|  2021-01-01 00:52:19|            1|         .20|        1|                N|
238|         151|           2|          3|  0.5|    0.5|         0|          0|                0.3|        4.3|
0|
|       1| 2021-01-01 00:43:30|  2021-01-01 01:11:06|            1|       14.70|        1|                N|
132|         165|           1|         42|  0.5|    0.5|      8.65|          0|                0.3|      51.95|
0|
+--------+--------------------+---------------------+---------------+-------------+----------+------------------+--------
----+------------+------------+-----------+-----+-------+----------+------------+--------------------+------------+----
----------------+
only showing top 3 rows
```