Clase 7

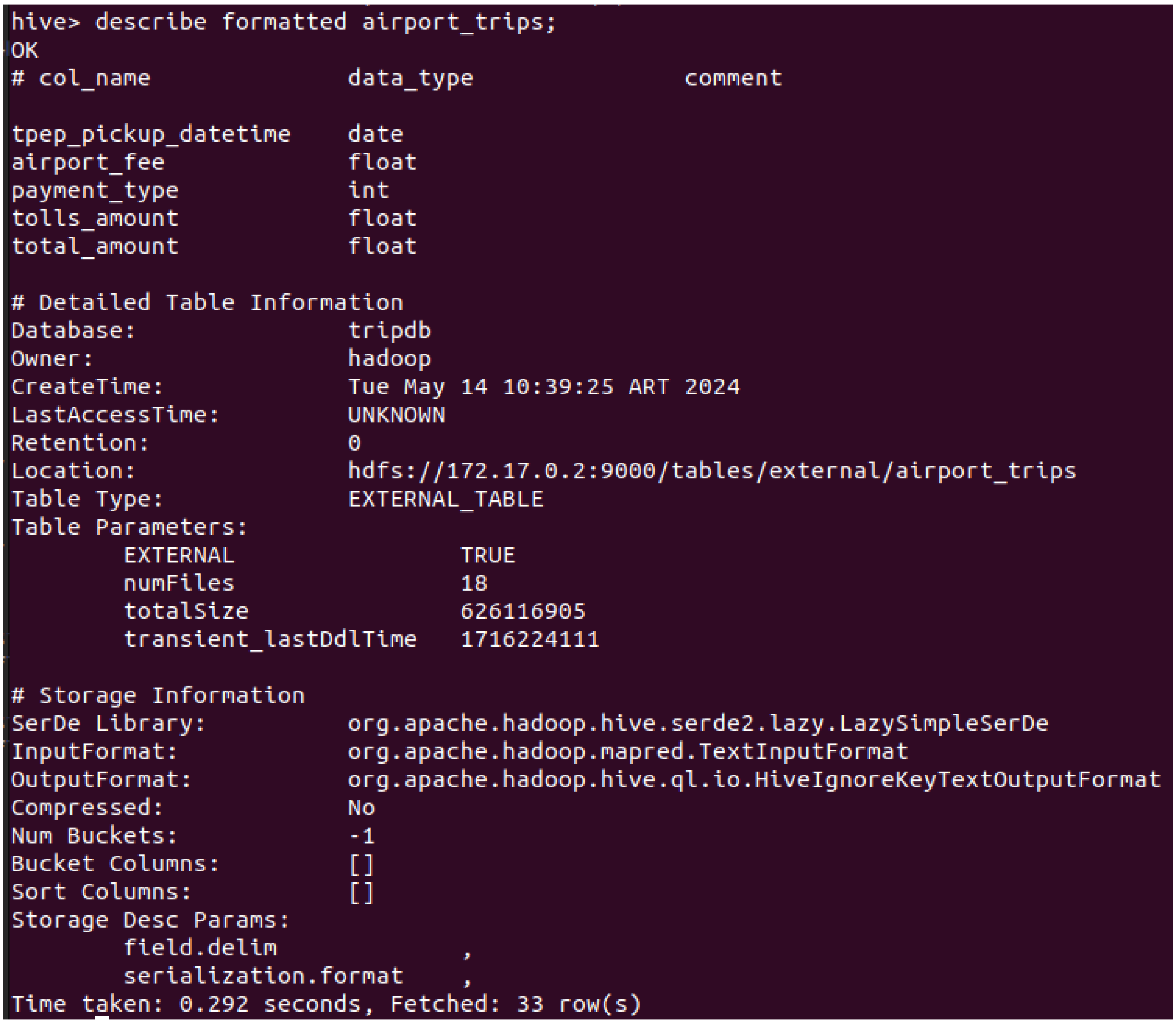
Diccionario de datos:
https://www.nyc.gov/assets/tlc/downloads/pdf/data_dictionary_trip_records_yellow.pdf

1. En Hive, crear la siguiente tabla (externa) en la base de datos tripdata:

airport_trips(tpep_pickup_datetetime, airport_fee, payment_type, tolls_amount, total_amount)

```
CREATE EXTERNAL TABLE airport_trips (tpep_pickup_datetime date, airport_fee float,
payment_type int, tolls_amount float, total_amount float)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LOCATION '/tables/external/airport_trips';
```

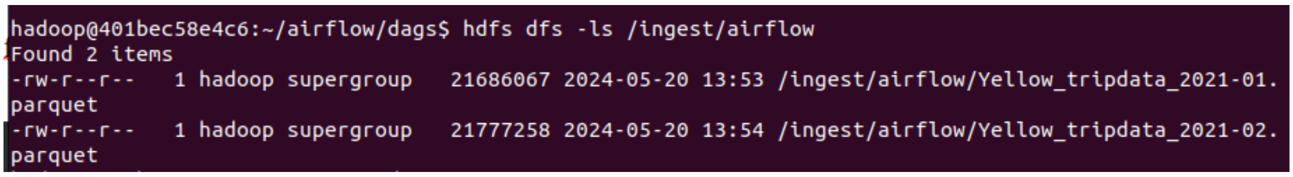
2. En Hive, mostrar el esquema de airport_trips



```
hive> describe formatted airport_trips;
OK
# col_name              data_type               comment

tpep_pickup_datetime    date
airport_fee             float
payment_type            int
tolls_amount            float
total_amount            float

# Detailed Table Information
Database:               tripdb
Owner:                  hadoop
CreateTime:             Tue May 14 10:39:25 ART 2024
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://172.17.0.2:9000/tables/external/airport_trips
Table Type:             EXTERNAL_TABLE
Table Parameters:
        EXTERNAL                TRUE
        numFiles                18
        totalSize               626116905
        transient_lastDdlTime   1716224111

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:             No
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
        field.delim             ,
        serialization.format    ,
Time taken: 0.292 seconds, Fetched: 33 row(s)
```

3. Crear un archivo .bash que permita descargar los archivos mencionados abajo e ingestarlos en HDFS:

```
wget -O /home/hadoop/landing/Yellow_tripdata_2021-01.parquet
https://edvaibucket.blob.core.windows.net/data-engineer-edvai/yellow_tripdata_2021-01.parquet\?
sp\=r\&st\=2023-11-06T12:52:39Z\&se\=2025-11-06T20:52:39Z\&sv\=2022-11-02\&sr\=c\&sig\
=J4Ddi2c7Ep23OhQLPisbYaerlH472iigPwc1%2FkG80EM%3D

/home/hadoop/hadoop/bin/hdfs dfs -put -f /home/hadoop/landing/Yellow_tripdata_2021-01.parquet
/ingest/airflow
```



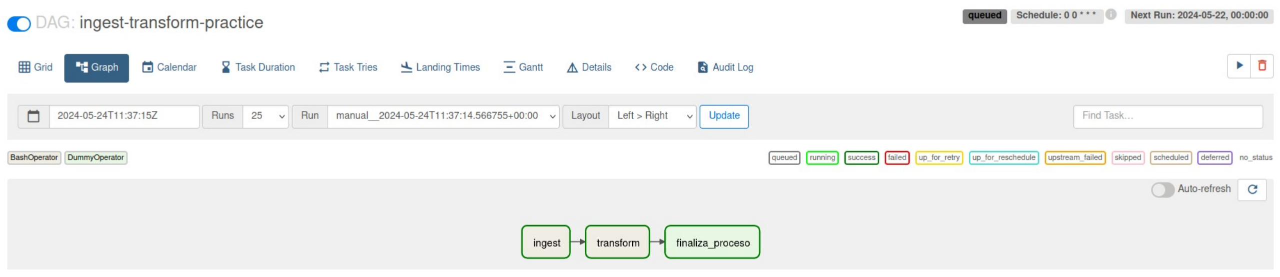
```
hadoop@401bec58e4c6:~/airflow/dags$ hdfs dfs -ls /ingest/airflow
Found 2 items
-rw-r--r--   1 hadoop supergroup   21686067 2024-05-20 13:53 /ingest/airflow/Yellow_tripdata_2021-01.
parquet
-rw-r--r--   1 hadoop supergroup   21777258 2024-05-20 13:54 /ingest/airflow/Yellow_tripdata_2021-02.
parquet
```

4. Crear un archivo .py que permita, mediante Spark, crear un data frame uniendo los viajes del mes 01 y mes 02 del año 2021 y luego Insertar en la tabla airport_trips los viajes que tuvieron como inicio o destino aeropuertos, que hayan pagado con dinero.

```python
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql import HiveContext

sc = SparkContext('local')
spark = SparkSession(sc)
hc = HiveContext(sc)

df01 = spark.read.option("header","true").parquet("hdfs://172.17.0.2:9000/ingest/airflow/
Yellow_tripdata_2021-01.parquet")

df02 = spark.read.option("header","true").parquet("hdfs://172.17.0.2:9000/ingest/airflow/
Yellow_tripdata_2021-02.parquet")

total_trips_df = df01.unionAll(df02)

total_trips_filtered_df = total_trips_df.filter("RatecodeID = 2 AND payment_type = 2")

final_total_trips_df =
total_trips_filtered_df.select(total_trips_filtered_df.tpep_pickup_datetime.cast("date"),
total_trips_filtered_df.airport_fee.cast("float"), total_trips_filtered_df.payment_type.cast("int"),
total_trips_filtered_df.tolls_amount.cast("float"), total_trips_filtered_df.total_amount.cast("float"))

#final_total_trips_df.show(10)

final_total_trips_df.createOrReplaceTempView("airport_trips_view")

hc.sql("insert into tripdb.airport_trips select * from airport_trips_view;")
```

5. Realizar un proceso automático en Airflow que orqueste los archivos creados en los puntos 3 y 4. Correrlo y mostrar una captura de pantalla (del DAG y del resultado en la base de datos)

```python
from datetime import timedelta
from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.dummy import DummyOperator
from airflow.utils.dates import days_ago

args = {
    'owner': 'airflow',
}

with DAG(
    dag_id='ingest-transform-practice',
    default_args=args,
    schedule_interval='0 0 * * *',
    start_date=days_ago(2),
    dagrun_timeout=timedelta(minutes=60),
    tags=['ingest', 'transform'],
    params={"example_key": "example_value"},
) as dag:

    finaliza_proceso = DummyOperator(
        task_id='finaliza_proceso',
    )
    ingest = BashOperator(
        task_id='ingest',
        bash_command='/usr/bin/sh /home/hadoop/scripts/practica_airflow/ingest.sh ',
    )
    transform = BashOperator(
        task_id='transform',
        bash_command='ssh hadoop@172.17.0.2 /home/hadoop/spark/bin/spark-submit --files
/home/hadoop/hive/conf/hive-site.xml /home/hadoop/scripts/practica_airflow/transformation.py ',
    )

    ingest >> transform >> finaliza_proceso

if __name__ == "__main__":
    dag.cli()
```



DAG: ingest-transform-practice

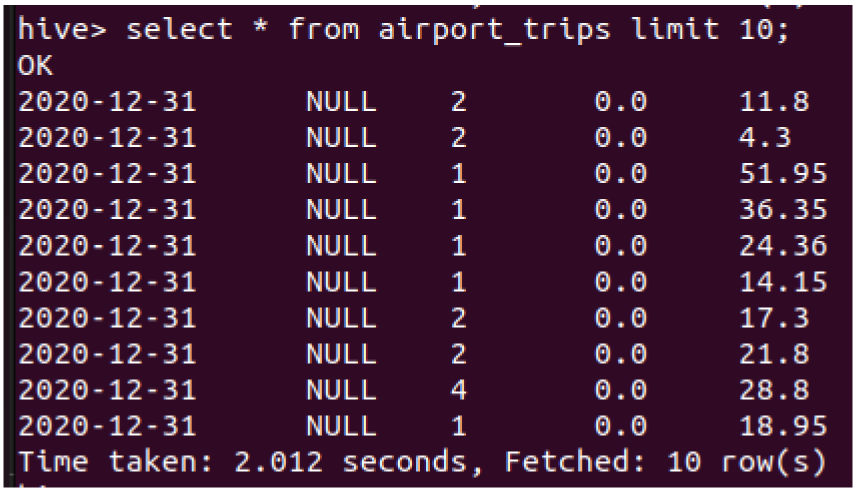
queued Schedule: 0 0 * * * Next Run: 2024-05-22, 00:00:00

2024-05-24T11:37:15Z Runs 25 Run manual__2024-05-24T11:37:14.566755+00:00 Layout Left > Right

Luego de ejecutarse el dag, en hive ya se han cargado los datos a la tabla airport_trips

```
hive> show databases;
OK
default
f1
tripdb
Time taken: 0.891 seconds, Fetched: 3 row(s)
```

```
hive> use tripdb;
OK
Time taken: 0.041 seconds

hive> show tables;
OK
airport_trips
congestion
distance
passenger
payments
tolls
Time taken: 0.088 seconds, Fetched: 6 row(s)

hive> select * from airport_trips limit 10;
```



```
hive> select * from airport_trips limit 10;
OK
2020-12-31      NULL    2       0.0     11.8
2020-12-31      NULL    2       0.0     4.3
2020-12-31      NULL    1       0.0     51.95
2020-12-31      NULL    1       0.0     36.35
2020-12-31      NULL    1       0.0     24.36
2020-12-31      NULL    1       0.0     14.15
2020-12-31      NULL    2       0.0     17.3
2020-12-31      NULL    2       0.0     21.8
2020-12-31      NULL    4       0.0     28.8
2020-12-31      NULL    1       0.0     18.95
Time taken: 2.012 seconds, Fetched: 10 row(s)
```

Clase 8

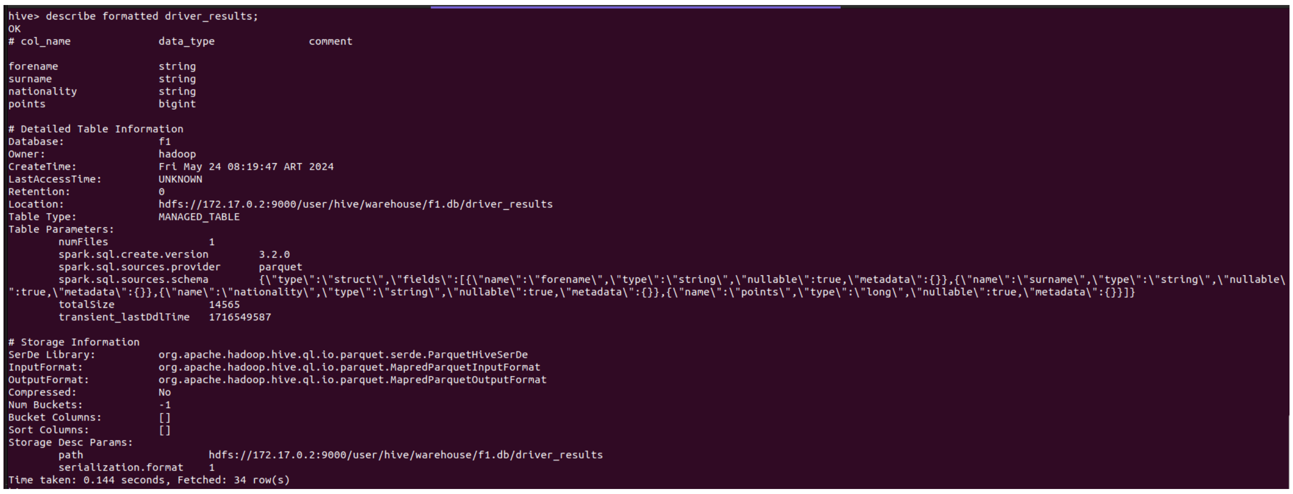
Diccionario de datos:
https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020?select=results.csv

1. Crear la siguientes tablas externas en la base de datos f1 en hive:

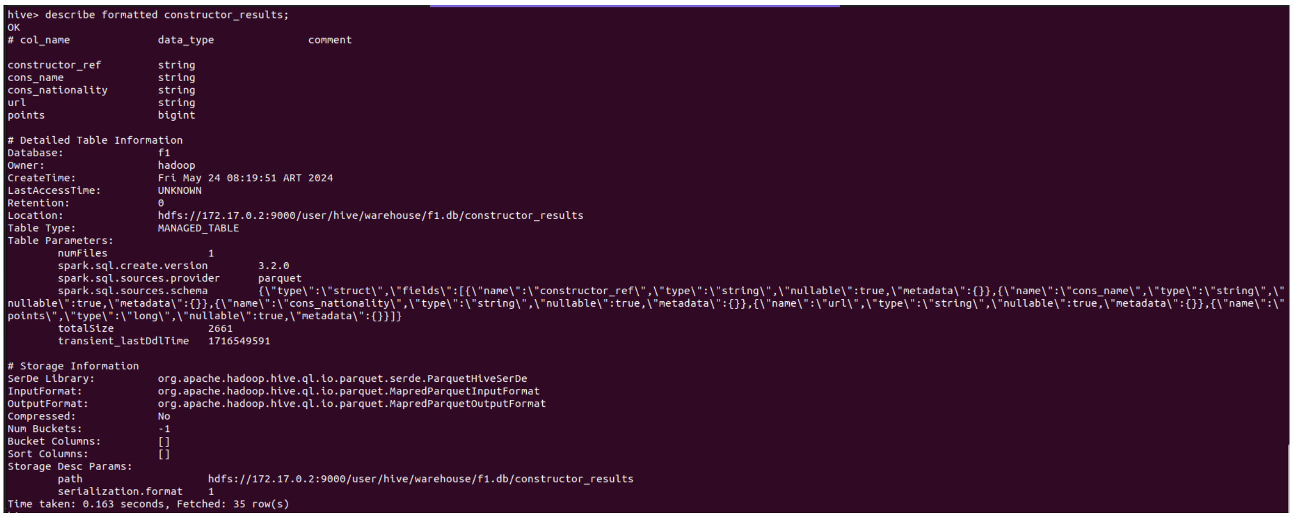
a. driver_results (driver_forename, driver_surname, driver_nationality, points)
b. constructor_results (constructorRef, cons_name, cons_nationality, url, points)

```
create external table driver_results(
driver_forename string,
driver_surename string,
driver_nationality string,
point int)
COMMENT 'Driver Results'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LOCATION '/tables/external/f1/';

create external table constructor_results(
constructor_ref string,
cons_name string,
cons_nationality string,
url string,
points int)
comment 'Constructor Results'
row format delimited
fields terminated by ','
location '/tables/external/f1/';
```

2. En Hive, mostrar el esquema de driver_results y constructor_results



```
hive> describe formatted driver_results;
OK
# col_name              data_type               comment

forename                string
surname                 string
nationality             string
points                  bigint

# Detailed Table Information
Database:               f1
Owner:                  hadoop
CreateTime:             Fri May 24 08:19:47 ART 2024
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://172.17.0.2:9000/user/hive/warehouse/f1.db/driver_results
Table Type:             MANAGED_TABLE
Table Parameters:
        numFiles                1
        spark.sql.create.version        3.2.0
        spark.sql.sources.provider      parquet
        spark.sql.sources.schema        {\"type\":\"struct\",\"fields\":[{\"name\":\"forename\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}},{\"name\":\"surname\",\"type\":\"string\",\"nullable\
":true,\"metadata\":{}},{\"name\":\"nationality\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}},{\"name\":\"points\",\"type\":\"long\",\"nullable\":true,\"metadata\":{}}]}
        totalSize               14565
        transient_lastDdlTime   1716549587

# Storage Information
SerDe Library:          org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe
InputFormat:            org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat
Compressed:             No
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
        path                    hdfs://172.17.0.2:9000/user/hive/warehouse/f1.db/driver_results
        serialization.format    1
Time taken: 0.144 seconds, Fetched: 34 row(s)
```



```
hive> describe formatted constructor_results;
OK
# col_name              data_type               comment

constructor_ref         string
cons_name               string
cons_nationality        string
url                     string
points                  bigint

# Detailed Table Information
Database:               f1
Owner:                  hadoop
CreateTime:             Fri May 24 08:19:51 ART 2024
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://172.17.0.2:9000/user/hive/warehouse/f1.db/constructor_results
Table Type:             MANAGED_TABLE
Table Parameters:
        numFiles                1
        spark.sql.create.version        3.2.0
        spark.sql.sources.provider      parquet
        spark.sql.sources.schema        {\"type\":\"struct\",\"fields\":[{\"name\":\"constructor_ref\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}},{\"name\":\"cons_name\",\"type\":\"string\",\"
nullable\":true,\"metadata\":{}},{\"name\":\"cons_nationality\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}},{\"name\":\"url\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}},{\"name\":\"
points\",\"type\":\"long\",\"nullable\":true,\"metadata\":{}}]}
        totalSize               2661
        transient_lastDdlTime   1716549591

# Storage Information
SerDe Library:          org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe
InputFormat:            org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat
Compressed:             No
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
        path                    hdfs://172.17.0.2:9000/user/hive/warehouse/f1.db/constructor_results
        serialization.format    1
Time taken: 0.163 seconds, Fetched: 35 row(s)
```

3. Crear un archivo .bash que permita descargar los archivos mencionados

```
wget -O /home/hadoop/landing/results.csv https://dataengineerpublic.blob.core.windows.net/data-engineer/f1/results.csv
hdfs dfs -put /home/hadoop/landing/results.csv /ingest/f1

wget -O /home/hadoop/landing/drivers.csv https://dataengineerpublic.blob.core.windows.net/data-engineer/f1/drivers.csv
hdfs dfs -put /home/hadoop/landing/drivers.csv /ingest/f1

wget -O /home/hadoop/landing/constructors.csv https://dataengineerpublic.blob.core.windows.net/data-engineer/f1/constructors.csv
hdfs dfs -put /home/hadoop/landing/constructors.csv /ingest/f1

wget -O /home/hadoop/landing/races.csv https://dataengineerpublic.blob.core.windows.net/data-engineer/f1/races.csv
hdfs dfs -put /home/hadoop/landing/races.csv /ingest/f1
```

4. Generar un archivo .py que permita, mediante Spark:

a. insertar en la tabla driver_results los corredores con mayor cantidad de puntos en la historia.

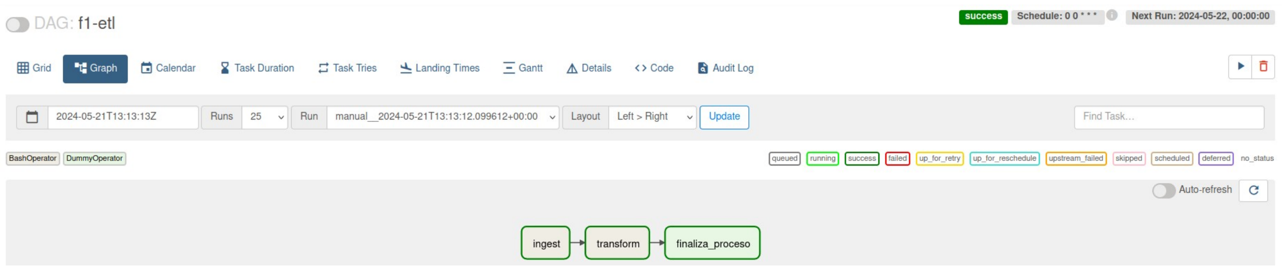
b. insertar en la tabla constructor_result quienes obtuvieron más puntos en el Spanish Grand Prix en el año 1991

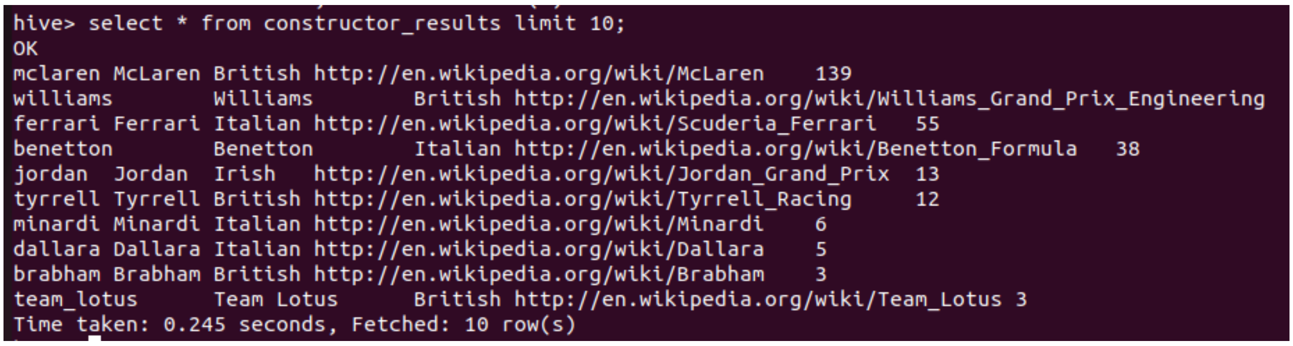
```python
from pyspark.sql.session import SparkSession

spark = SparkSession.builder \
        .appName("CSV to Hive") \
        .config("spark.sql.catalog.Implementation", "hive") \
        .enableHiveSupport() \
        .getOrCreate()

df_results = spark.read.option("header", "true").csv("hdfs://172.17.0.2:9000/ingest/f1/results.csv")
df_drivers = spark.read.option("header", "true").csv("hdfs://172.17.0.2:9000/ingest/f1/drivers.csv")
df_constructors = spark.read.option("header", "true").csv("hdfs://172.17.0.2:9000/ingest/f1/constructors.csv")
df_races = spark.read.option("header", "true").csv("hdfs://172.17.0.2:9000/ingest/f1/races.csv")
```

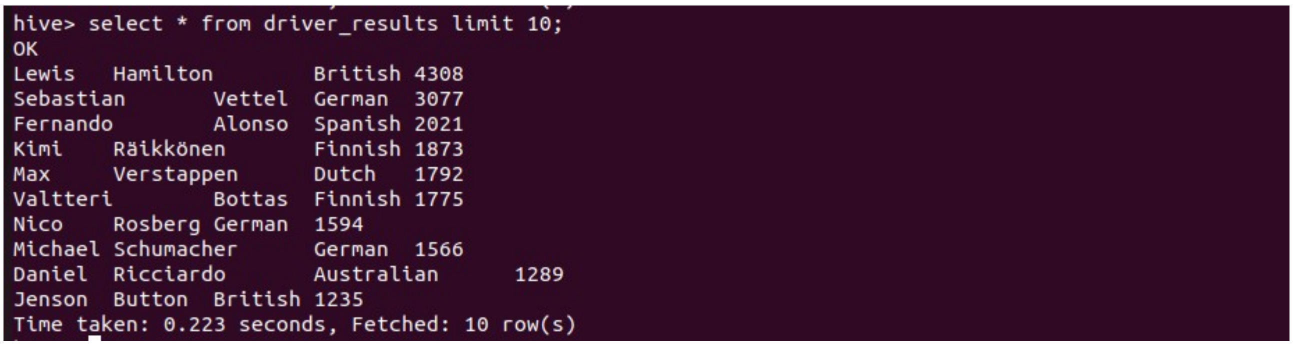
```
df_results.createOrReplaceTempView("vw_results")
df_drivers.createOrReplaceTempView("vw_drivers")
df_constructors.createOrReplaceTempView("vw_constructors")
df_races.createOrReplaceTempView("vw_races")

df_driver_results = spark.sql( \
	"SELECT \
		cast(d.forename as string), \
		cast(d.surname as string), \
		cast(d.nationality as string), \
		sum(cast(r.points as int)) points \
	FROM \
		vw_results r, \
		vw_drivers d \
	WHERE \
		r.driverId = d.driverId \
	GROUP BY \
		d.driverId, d.forename, d.surname, d.nationality \
	ORDER BY \
		points DESC" \
	)

df_constructor_results = spark.sql( \
	"SELECT \
		cast(c.constructorRef as string) constructor_ref, \
		cast(c.name as string) cons_name, \
		cast(c.nationality as string) cons_nationality, \
		cast(c.url as string) url, \
		sum(cast(r.points as int)) points \
	FROM \
		vw_results r, \
		vw_constructors c, \
		vw_races ra \
	WHERE \
		r.constructorId = c.constructorId \
		AND r.raceId = ra.raceId \
		AND ra.year = '1991' \
	GROUP BY \
		c.constructorId, constructor_ref, cons_name, cons_nationality, c.url \
	ORDER BY \
		points DESC" \
	)

df_driver_results.write.mode("overwrite").saveAsTable("f1.driver_results")
df_constructor_results.write.mode("overwrite").saveAsTable("f1.constructor_results")
```

5. Realizar un proceso automático en Airflow que orqueste los archivos creados en los puntos 3 y 4. Correrlo y mostrar una captura de pantalla (del DAG y del resultado en la base de datos)

```python
from datetime import timedelta
from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.dummy import DummyOperator
from airflow.utils.dates import days_ago

args = {
    'owner': 'airflow',
}

with DAG(
    dag_id='f1-etl',
    default_args=args,
    schedule_interval='0 0 * * *',
    start_date=days_ago(2),
    dagrun_timeout=timedelta(minutes=60),
    tags=['ingest', 'transform'],
    params={"example_key": "example_value"},

) as dag:

    finaliza_proceso = DummyOperator(
        task_id='finaliza_proceso',
    )
    ingest = BashOperator(
        task_id='ingest',
        bash_command='/usr/bin/sh
/home/hadoop/scripts/practica_airflow/clase8/ingest.sh ',
    )
    transform = BashOperator(
        task_id='transform',
        bash_command='ssh hadoop@172.17.0.2 /home/hadoop/spark/bin/spark-submit
--files /home/hadoop/hive/conf/hive-site.xml
/home/hadoop/scripts/practica_airflow/f1-transformation2.py ',
    )

    ingest >> transform >> finaliza_proceso


if __name__ == "__main__":
    dag.cli()
```


DAG: f1-etl
success Schedule: 0 0 * * * Next Run: 2024-05-22, 00:00:00
2024-05-21T13:13:13Z Runs 25 Run manual__2024-05-21T13:13:12.099612+00:00 Layout Left > Right


```
hive> select * from constructor_results limit 10;
OK
mclaren McLaren British http://en.wikipedia.org/wiki/McLaren    139
williams        Williams        British http://en.wikipedia.org/wiki/Williams_Grand_Prix_Engineering
ferrari Ferrari Italian http://en.wikipedia.org/wiki/Scuderia_Ferrari    55
benetton        Benetton        Italian http://en.wikipedia.org/wiki/Benetton_Formula    38
jordan  Jordan  Irish   http://en.wikipedia.org/wiki/Jordan_Grand_Prix  13
tyrrell Tyrrell British http://en.wikipedia.org/wiki/Tyrrell_Racing     12
minardi Minardi Italian http://en.wikipedia.org/wiki/Minardi    6
dallara Dallara Italian http://en.wikipedia.org/wiki/Dallara    5
brabham Brabham British http://en.wikipedia.org/wiki/Brabham    3
team_lotus      Team Lotus      British http://en.wikipedia.org/wiki/Team_Lotus 3
Time taken: 0.245 seconds, Fetched: 10 row(s)
```

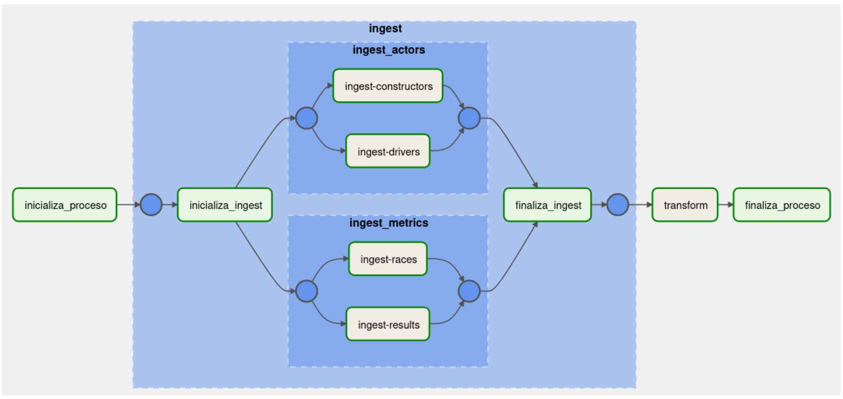


```
hive> select * from driver_results limit 10;
OK
Lewis   Hamilton        British 4308
Sebastian       Vettel  German  3077
Fernando        Alonso  Spanish 2021
Kimi    Räikkönen       Finnish 1873
Max     Verstappen      Dutch   1792
Valtteri        Bottas  Finnish 1775
Nico    Rosberg German  1594
Michael Schumacher      German  1566
Daniel  Ricciardo       Australian      1289
Jenson  Button  British 1235
Time taken: 0.223 seconds, Fetched: 10 row(s)
```

BONUS: GroupTask

En lugar de la tarea de ingest anterior que contenía todas las sentencias para la descarga y puesta en hdfs de los archivos, se separó por dos task group donde en una se ejecutan los comandos para los archivos de conductores y constructores, mientras que en el otro grupo se ejecutan las tareas para los archivos de carreras y resultados.

```
with TaskGroup("ingest") as ingest:

      inicializa_ingest = EmptyOperator(task_id='inicializa_ingest',)

      finaliza_ingest = EmptyOperator(task_id='finaliza_ingest',)

      with TaskGroup("ingest_actors") as ingest_actors:

        ingestDrivers = BashOperator(
        task_id='ingest-drivers',
        bash_command='/usr/bin/sh/home/hadoop/scripts/practica_airflow/clase8/
ingest-drivers.sh ',
        )

        ingestConstructors = BashOperator(
        task_id='ingest-constructors',
        bash_command='/usr/bin/sh/home/hadoop/scripts/practica_airflow/clase8/
ingest-constructors.sh ',
        )

      with TaskGroup("ingest_metrics") as ingest_metrics:

        ingestResults = BashOperator(
        task_id='ingest-results',
        bash_command='/usr/bin/sh/home/hadoop/scripts/practica_airflow/clase8/
ingest-results.sh ',
        )

        ingestRaces = BashOperator(
        task_id='ingest-races',
        bash_command='/usr/bin/sh/home/hadoop/scripts/practica_airflow/clase8/
ingest-races.sh ',
        )

      inicializa_ingest >> [ingest_actors, ingest_metrics] >> finaliza_ingest

inicializa_proceso >>  ingest >> transform >> finaliza_proceso
```


ingest
ingest_actors
ingest-constructors
ingest-drivers
inicializa_proceso
inicializa_ingest
finaliza_ingest
transform
finaliza_proceso
ingest_metrics
ingest-races
ingest-results