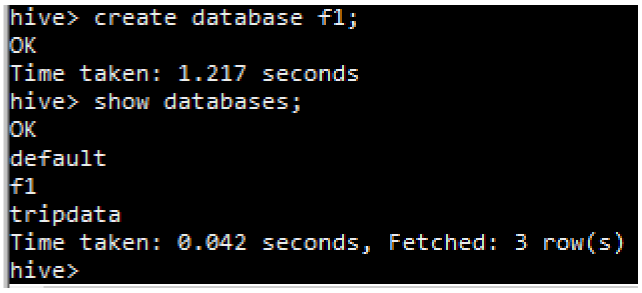
Clase 8

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)

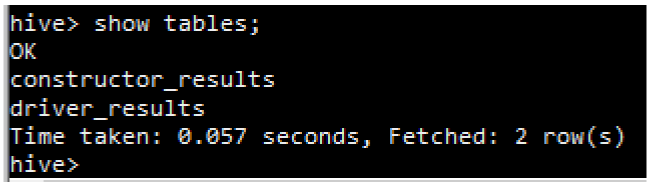
CREATE DATABASE f1;



```
hive> create database f1;
OK
Time taken: 1.217 seconds
hive> show databases;
OK
default
f1
tripdata
Time taken: 0.042 seconds, Fetched: 3 row(s)
hive>
```

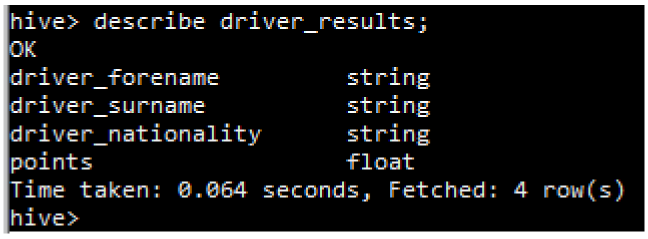
CREATE EXTERNAL TABLE f1.driver_results(driver_forename string, driver_surname string, driver_nationality string, points float)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LOCATION '/tables/external/f1/driver_results';

CREATE EXTERNAL TABLE f1.constructor_results(constructorRef string, cons_name string, cons_nationality string, url string, points float)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LOCATION '/tables/external/f1/constructor_results';

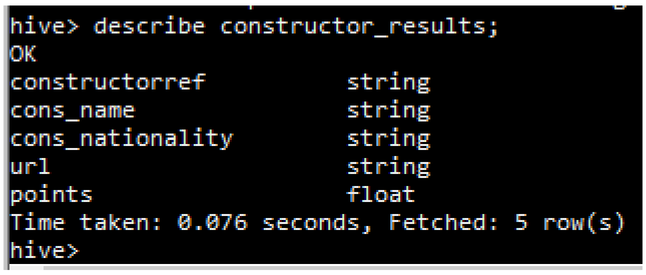


```
hive> show tables;
OK
constructor_results
driver_results
Time taken: 0.057 seconds, Fetched: 2 row(s)
hive>
```

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

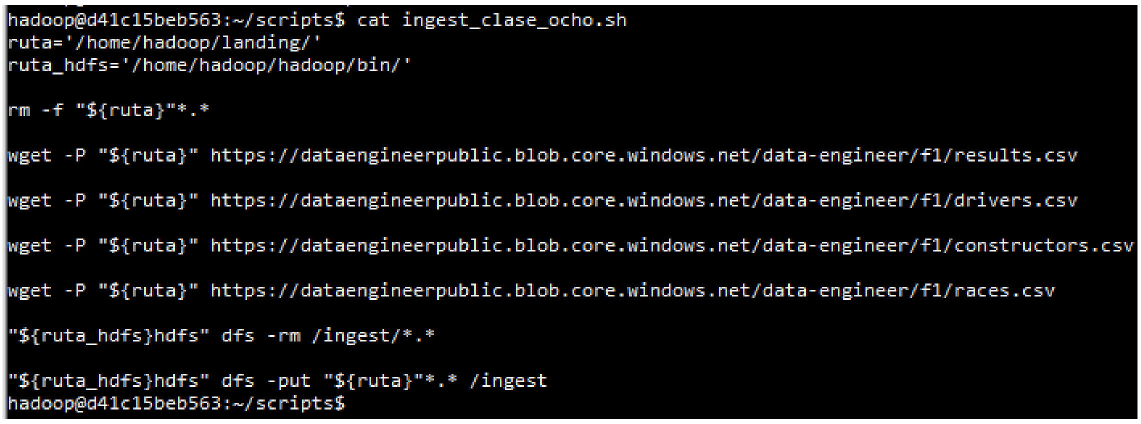


```
hive> describe driver_results;
OK
driver_forename         string
driver_surname          string
driver_nationality      string
points                  float
Time taken: 0.064 seconds, Fetched: 4 row(s)
hive>
```



```
hive> describe constructor_results;
OK
constructorref          string
cons_name               string
cons_nationality        string
url                     string
points                  float
Time taken: 0.076 seconds, Fetched: 5 row(s)
hive>
```

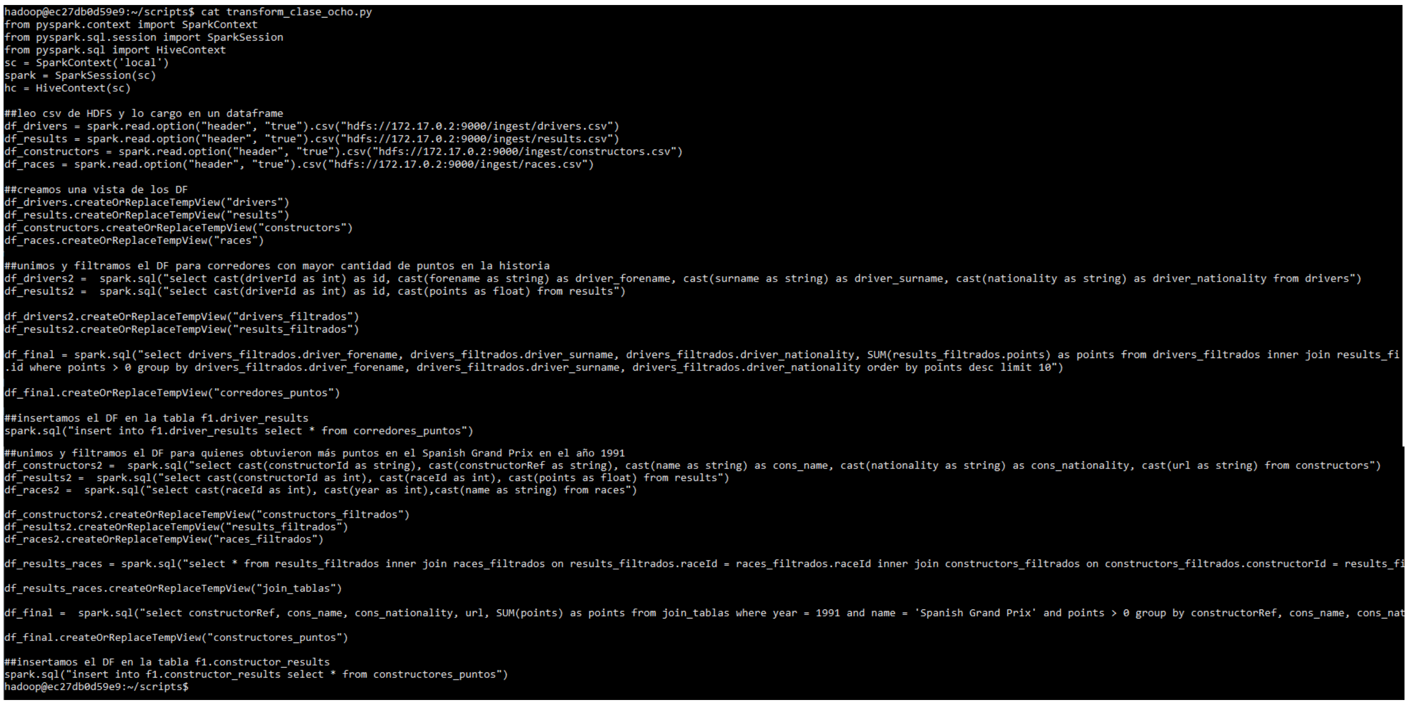
3. Crear un archivo .bash que permita descargar los archivos mencionados abajo e ingestarlos en HDFS:
results.csv
https://dataengineerpublic.blob.core.windows.net/data-engineer/f1/results.csv
drivers.csv
https://dataengineerpublic.blob.core.windows.net/data-engineer/f1/drivers.csv
constructors.csv
https://dataengineerpublic.blob.core.windows.net/data-engineer/f1/constructors.csv
races.csv
https://dataengineerpublic.blob.core.windows.net/data-engineer/f1/races.csv



```
hadoop@d41c15beb563:~/scripts$ cat ingest_clase_ocho.sh
ruta='/home/hadoop/landing/'
ruta_hdfs='/home/hadoop/hadoop/bin/'

rm -f "${ruta}"*.*

wget -P "${ruta}" https://dataengineerpublic.blob.core.windows.net/data-engineer/f1/results.csv

wget -P "${ruta}" https://dataengineerpublic.blob.core.windows.net/data-engineer/f1/drivers.csv

wget -P "${ruta}" https://dataengineerpublic.blob.core.windows.net/data-engineer/f1/constructors.csv

wget -P "${ruta}" https://dataengineerpublic.blob.core.windows.net/data-engineer/f1/races.csv

"${ruta_hdfs}hdfs" dfs -rm /ingest/*.*

"${ruta_hdfs}hdfs" dfs -put "${ruta}"*.* /ingest
hadoop@d41c15beb563:~/scripts$
```

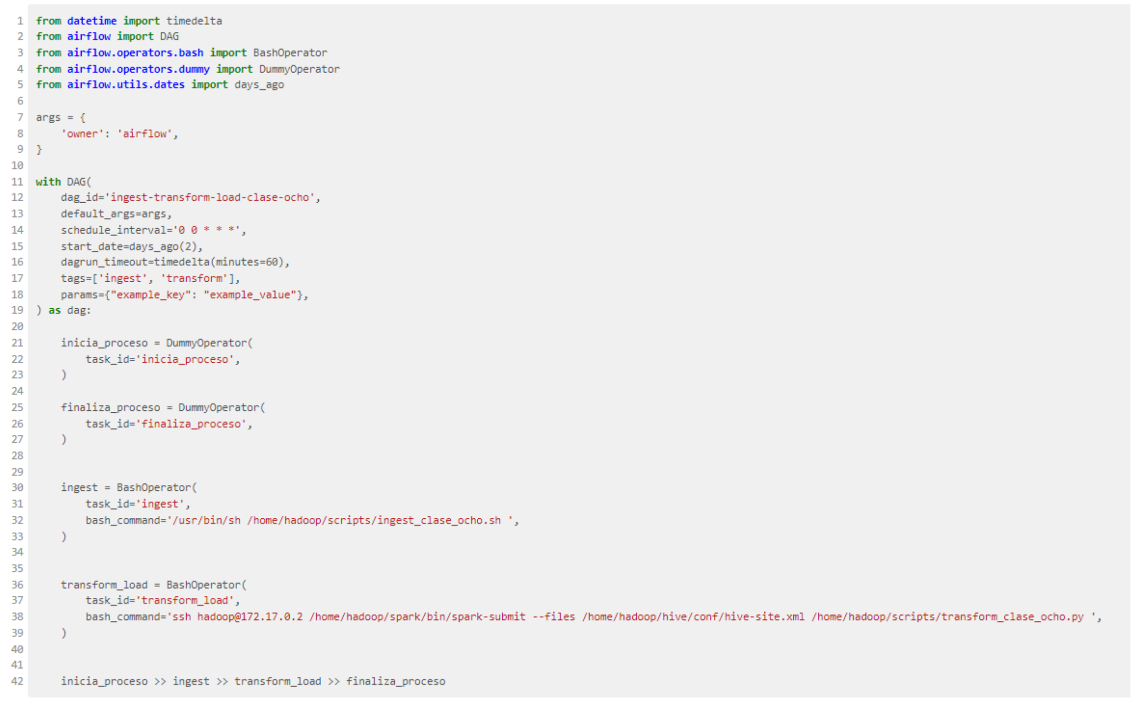
4. Generar un archivo .py que permita, mediante Spark:

- **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.**

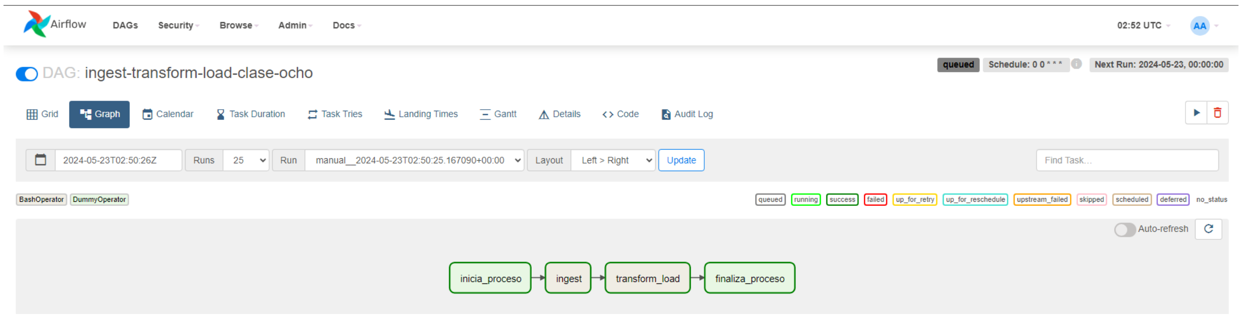


```
hadoop@ec27db0d59e9:~/scripts$ cat transform_clase_ocho.py
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql import HiveContext
sc = SparkContext('local')
spark = SparkSession(sc)
hc = HiveContext(sc)

##leo csv de HDFS y lo cargo en un dataframe
df_drivers = spark.read.option("header", "true").csv("hdfs://172.17.0.2:9000/ingest/drivers.csv")
df_results = spark.read.option("header", "true").csv("hdfs://172.17.0.2:9000/ingest/results.csv")
df_constructors = spark.read.option("header", "true").csv("hdfs://172.17.0.2:9000/ingest/constructors.csv")
df_races = spark.read.option("header", "true").csv("hdfs://172.17.0.2:9000/ingest/races.csv")

##creamos una vista de los DF
df_drivers.createOrReplaceTempView("drivers")
df_results.createOrReplaceTempView("results")
df_constructors.createOrReplaceTempView("constructors")
df_races.createOrReplaceTempView("races")

##unimos y filtramos el DF para corredores con mayor cantidad de puntos en la historia
df_drivers2 =  spark.sql("select cast(driverId as int) as id, cast(forename as string) as driver_forename, cast(surname as string) as driver_surname, cast(nationality as string) as driver_nationality from drivers")
df_results2 =  spark.sql("select cast(driverId as int) as id, cast(points as float) from results")

df_drivers2.createOrReplaceTempView("drivers_filtrados")
df_results2.createOrReplaceTempView("results_filtrados")

df_final = spark.sql("select drivers_filtrados.driver_forename, drivers_filtrados.driver_surname, drivers_filtrados.driver_nationality, SUM(results_filtrados.points) as points from drivers_filtrados inner join results_fi
.id where points > 0 group by drivers_filtrados.driver_forename, drivers_filtrados.driver_surname, drivers_filtrados.driver_nationality order by points desc limit 10")

df_final.createOrReplaceTempView("corredores_puntos")

##insertamos el DF en la tabla f1.driver_results
spark.sql("insert into f1.driver_results select * from corredores_puntos")

##unimos y filtramos el DF para quienes obtuvieron más puntos en el Spanish Grand Prix en el año 1991
df_constructors2 =  spark.sql("select cast(constructorId as string), cast(constructorRef as string), cast(name as string) as cons_name, cast(nationality as string) as cons_nationality, cast(url as string) from constructors")
df_results2 =  spark.sql("select cast(constructorId as int), cast(raceId as int), cast(points as float) from results")
df_races2 =  spark.sql("select cast(raceId as int), cast(year as int),cast(name as string) from races")

df_constructors2.createOrReplaceTempView("constructors_filtrados")
df_results2.createOrReplaceTempView("results_filtrados")
df_races2.createOrReplaceTempView("races_filtrados")

df_results_races = spark.sql("select * from results_filtrados inner join races_filtrados on results_filtrados.raceId = races_filtrados.raceId inner join constructors_filtrados on constructors_filtrados.constructorId = results_fi

df_results_races.createOrReplaceTempView("join_tablas")

df_final =  spark.sql("select constructorRef, cons_name, cons_nationality, url, SUM(points) as points from join_tablas where year = 1991 and name = 'Spanish Grand Prix' and points > 0 group by constructorRef, cons_name, cons_nat

df_final.createOrReplaceTempView("constructores_puntos")

##insertamos el DF en la tabla f1.constructor_results
spark.sql("insert into f1.constructor_results select * from constructores_puntos")
hadoop@ec27db0d59e9:~/scripts$
```

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)



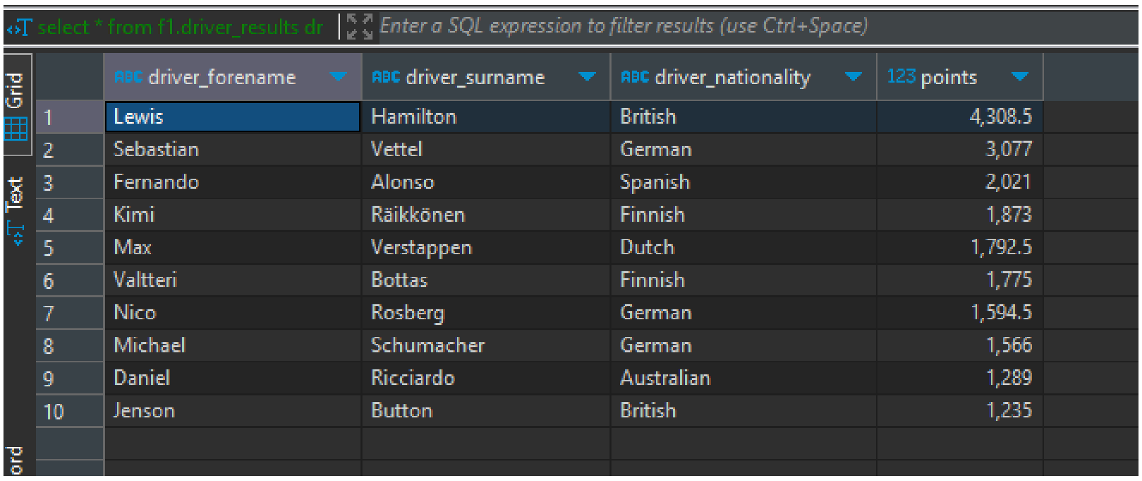
```python
from datetime import timedelta
from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.dummy import DummyOperator
from airflow.utils.dates import days_ago

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

with DAG(
    dag_id='ingest-transform-load-clase-ocho',
    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:

    inicia_proceso = DummyOperator(
        task_id='inicia_proceso',
    )

    finaliza_proceso = DummyOperator(
        task_id='finaliza_proceso',
    )


    ingest = BashOperator(
        task_id='ingest',
        bash_command='/usr/bin/sh /home/hadoop/scripts/ingest_clase_ocho.sh ',
    )


    transform_load = BashOperator(
        task_id='transform_load',
        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/transform_clase_ocho.py ',
    )


    inicia_proceso >> ingest >> transform_load >> finaliza_proceso
```



DAG: ingest-transform-load-clase-ocho

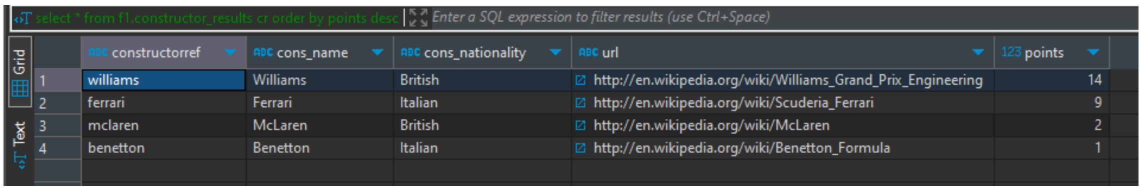
queued | Schedule: 0 0 * * * | Next Run: 2024-05-23, 00:00:00

2024-05-23T02:50:26Z | Runs 25 | Run manual__2024-05-23T02:50:25.167090+00:00 | Layout Left > Right



select * from f1.driver_results dr

	driver_forename	driver_surname	driver_nationality	points
1	Lewis	Hamilton	British	4,308.5
2	Sebastian	Vettel	German	3,077
3	Fernando	Alonso	Spanish	2,021
4	Kimi	Räikkönen	Finnish	1,873
5	Max	Verstappen	Dutch	1,792.5
6	Valtteri	Bottas	Finnish	1,775
7	Nico	Rosberg	German	1,594.5
8	Michael	Schumacher	German	1,566
9	Daniel	Ricciardo	Australian	1,289
10	Jenson	Button	British	1,235



```
select * from f1.constructor_results cr order by points desc
```

	constructorref	cons_name	cons_nationality	url	points
1	williams	Williams	British	http://en.wikipedia.org/wiki/Williams_Grand_Prix_Engineering	14
2	ferrari	Ferrari	Italian	http://en.wikipedia.org/wiki/Scuderia_Ferrari	9
3	mclaren	McLaren	British	http://en.wikipedia.org/wiki/McLaren	2
4	benetton	Benetton	Italian	http://en.wikipedia.org/wiki/Benetton_Formula	1