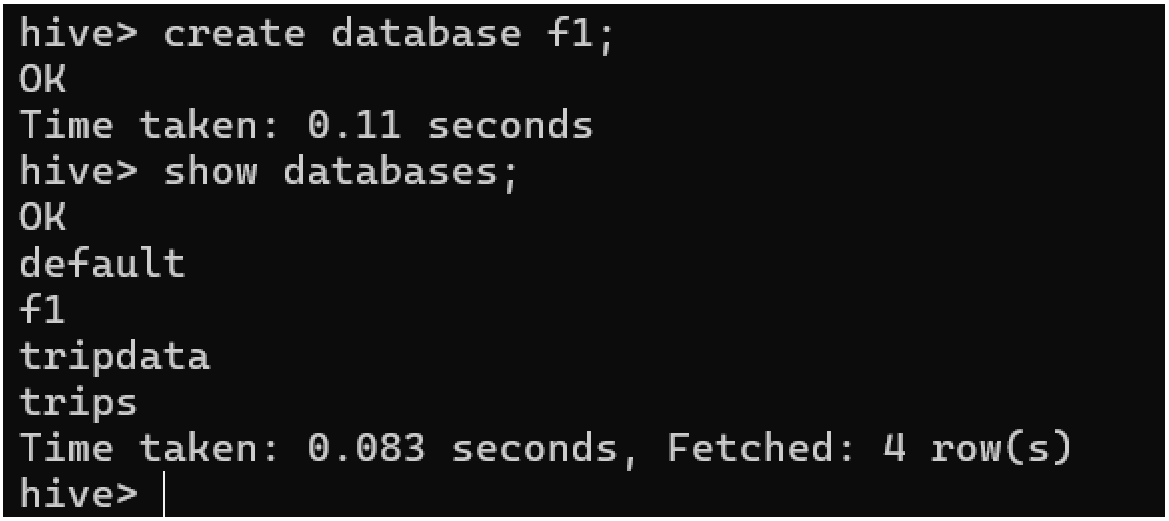
Clase 8

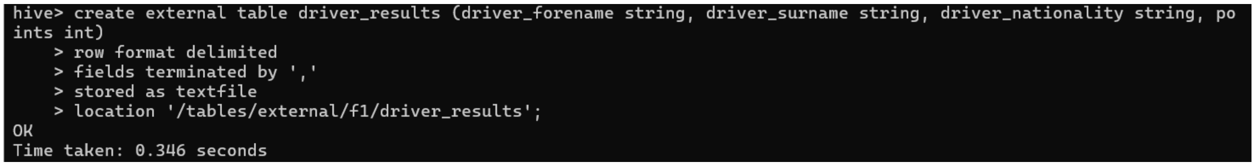
Consigna: Por cada ejercicio, escribir el código y agregar una captura de pantalla del resultado obtenido.

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

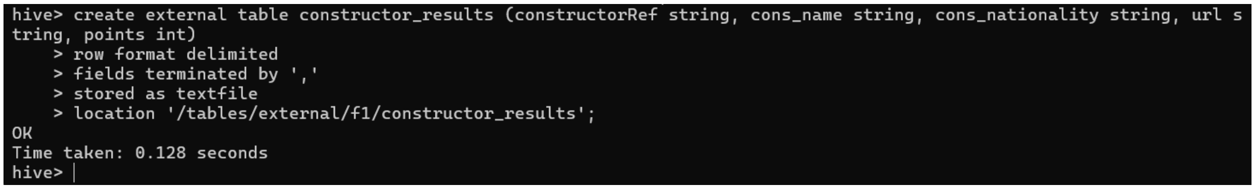
1. Crear la siguientes tablas externas en la base de datos f1 en hive:
    a. driver_results (driver_forename, driver_surname, driver_nationality, points)
    b. constructor_results (constructorRef, cons_name, cons_nationality, url, points)



```
hive> create database f1;
OK
Time taken: 0.11 seconds
hive> show databases;
OK
default
f1
tripdata
trips
Time taken: 0.083 seconds, Fetched: 4 row(s)
hive>
```

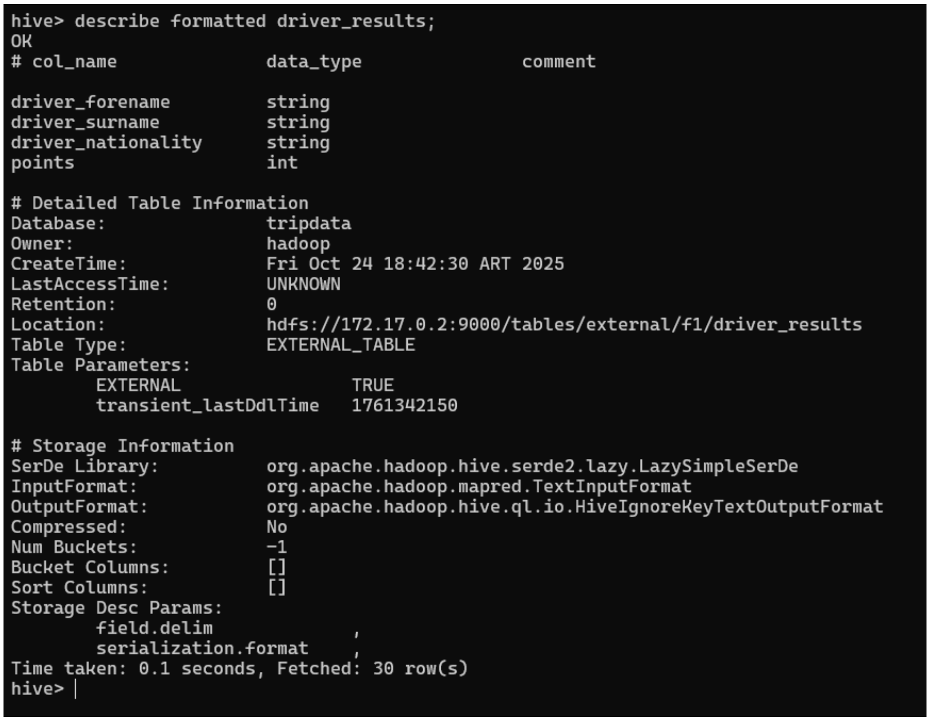


```
hive> create external table driver_results (driver_forename string, driver_surname string, driver_nationality string, po
ints int)
    > row format delimited
    > fields terminated by ','
    > stored as textfile
    > location '/tables/external/f1/driver_results';
OK
Time taken: 0.346 seconds
```

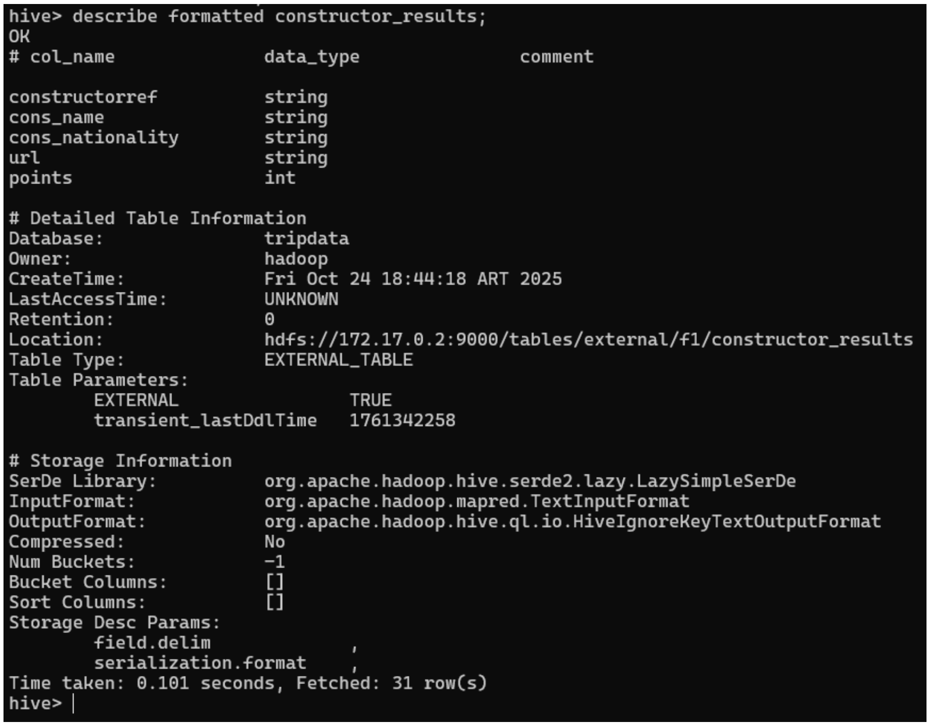


```
hive> create external table constructor_results (constructorRef string, cons_name string, cons_nationality string, url s
tring, points int)
    > row format delimited
    > fields terminated by ','
    > stored as textfile
    > location '/tables/external/f1/constructor_results';
OK
Time taken: 0.128 seconds
hive>
```

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



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

driver_forename         string
driver_surname          string
driver_nationality      string
points                  int

# Detailed Table Information
Database:               tripdata
Owner:                  hadoop
CreateTime:             Fri Oct 24 18:42:30 ART 2025
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://172.17.0.2:9000/tables/external/f1/driver_results
Table Type:             EXTERNAL_TABLE
Table Parameters:
        EXTERNAL                TRUE
        transient_lastDdlTime   1761342150

# 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: 0.1 seconds, Fetched: 30 row(s)
hive>
```



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

constructorref          string
cons_name               string
cons_nationality        string
url                     string
points                  int

# Detailed Table Information
Database:               tripdata
Owner:                  hadoop
CreateTime:             Fri Oct 24 18:44:18 ART 2025
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://172.17.0.2:9000/tables/external/f1/constructor_results
Table Type:             EXTERNAL_TABLE
Table Parameters:
        EXTERNAL                TRUE
        transient_lastDdlTime   1761342258

# 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: 0.101 seconds, Fetched: 31 row(s)
hive>
```

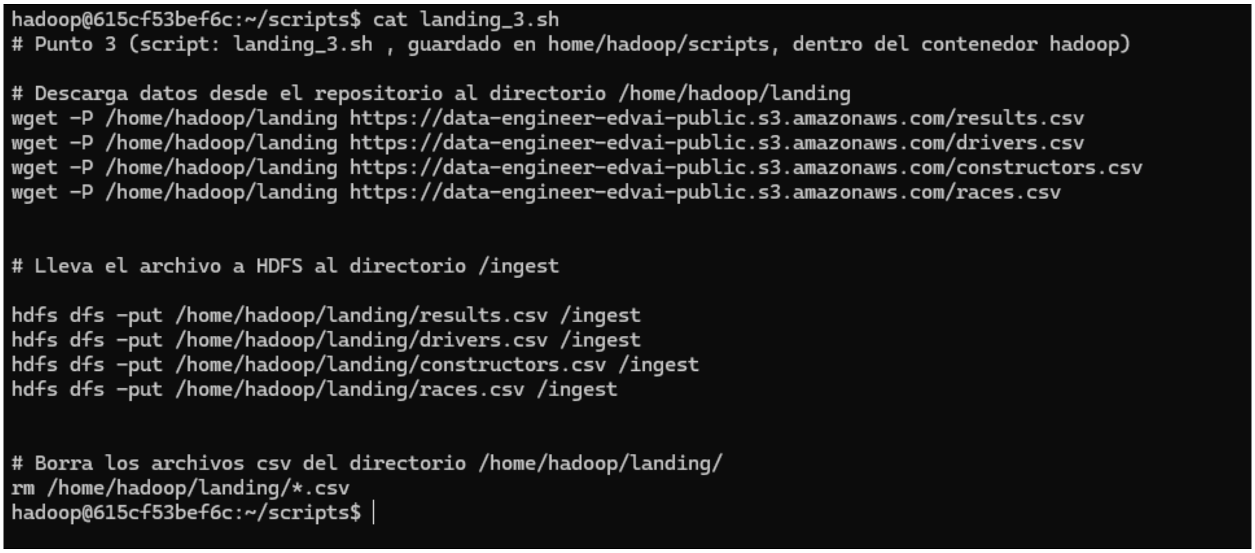
3. Crear un archivo .bash que permita descargar los archivos mencionados abajo e ingestarlos en HDFS:

    results.csv
    https://data-engineer-edvai-public.s3.amazonaws.com/results.csv

    drivers.csv
    https://data-engineer-edvai-public.s3.amazonaws.com/drivers.csv

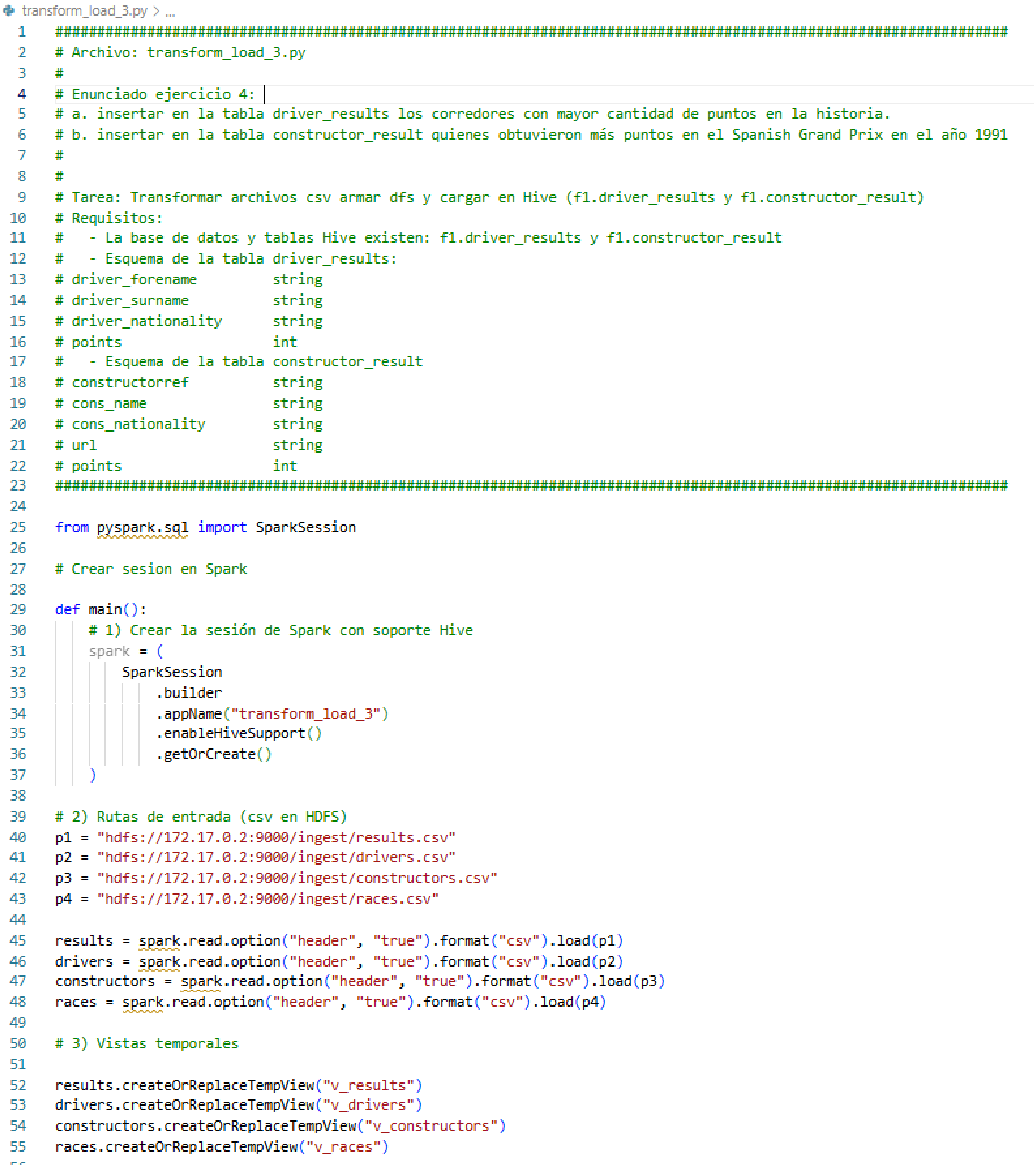
    constructors.csv
    https://data-engineer-edvai-public.s3.amazonaws.com/constructors.csv

    races.csv
    https://data-engineer-edvai-public.s3.amazonaws.com/races.csv



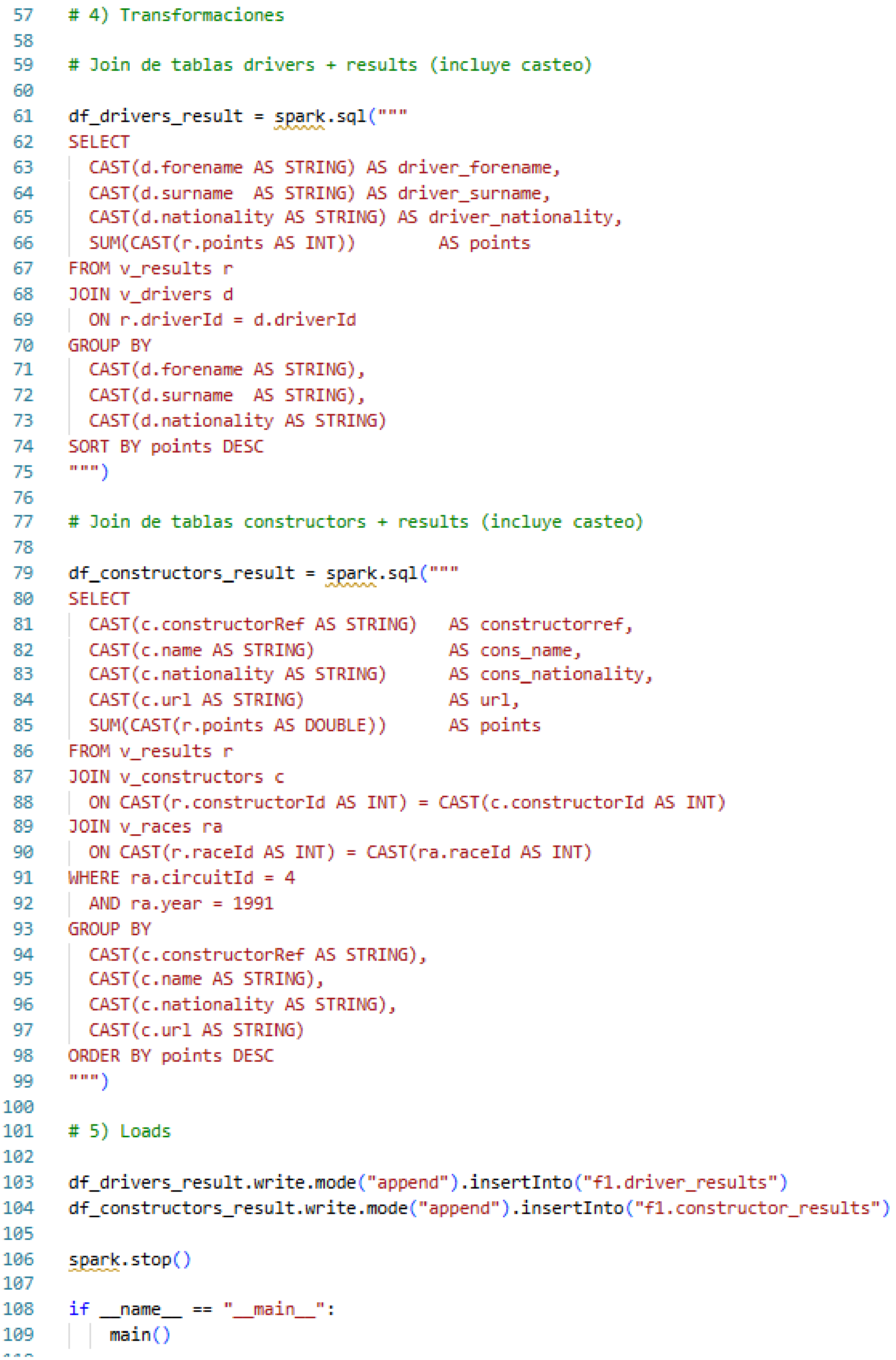
```
hadoop@615cf53bef6c:~/scripts$ cat landing_3.sh
# Punto 3 (script: landing_3.sh , guardado en home/hadoop/scripts, dentro del contenedor hadoop)

# Descarga datos desde el repositorio al directorio /home/hadoop/landing
wget -P /home/hadoop/landing https://data-engineer-edvai-public.s3.amazonaws.com/results.csv
wget -P /home/hadoop/landing https://data-engineer-edvai-public.s3.amazonaws.com/drivers.csv
wget -P /home/hadoop/landing https://data-engineer-edvai-public.s3.amazonaws.com/constructors.csv
wget -P /home/hadoop/landing https://data-engineer-edvai-public.s3.amazonaws.com/races.csv


# Lleva el archivo a HDFS al directorio /ingest

hdfs dfs -put /home/hadoop/landing/results.csv /ingest
hdfs dfs -put /home/hadoop/landing/drivers.csv /ingest
hdfs dfs -put /home/hadoop/landing/constructors.csv /ingest
hdfs dfs -put /home/hadoop/landing/races.csv /ingest


# Borra los archivos csv del directorio /home/hadoop/landing/
rm /home/hadoop/landing/*.csv
hadoop@615cf53bef6c:~/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



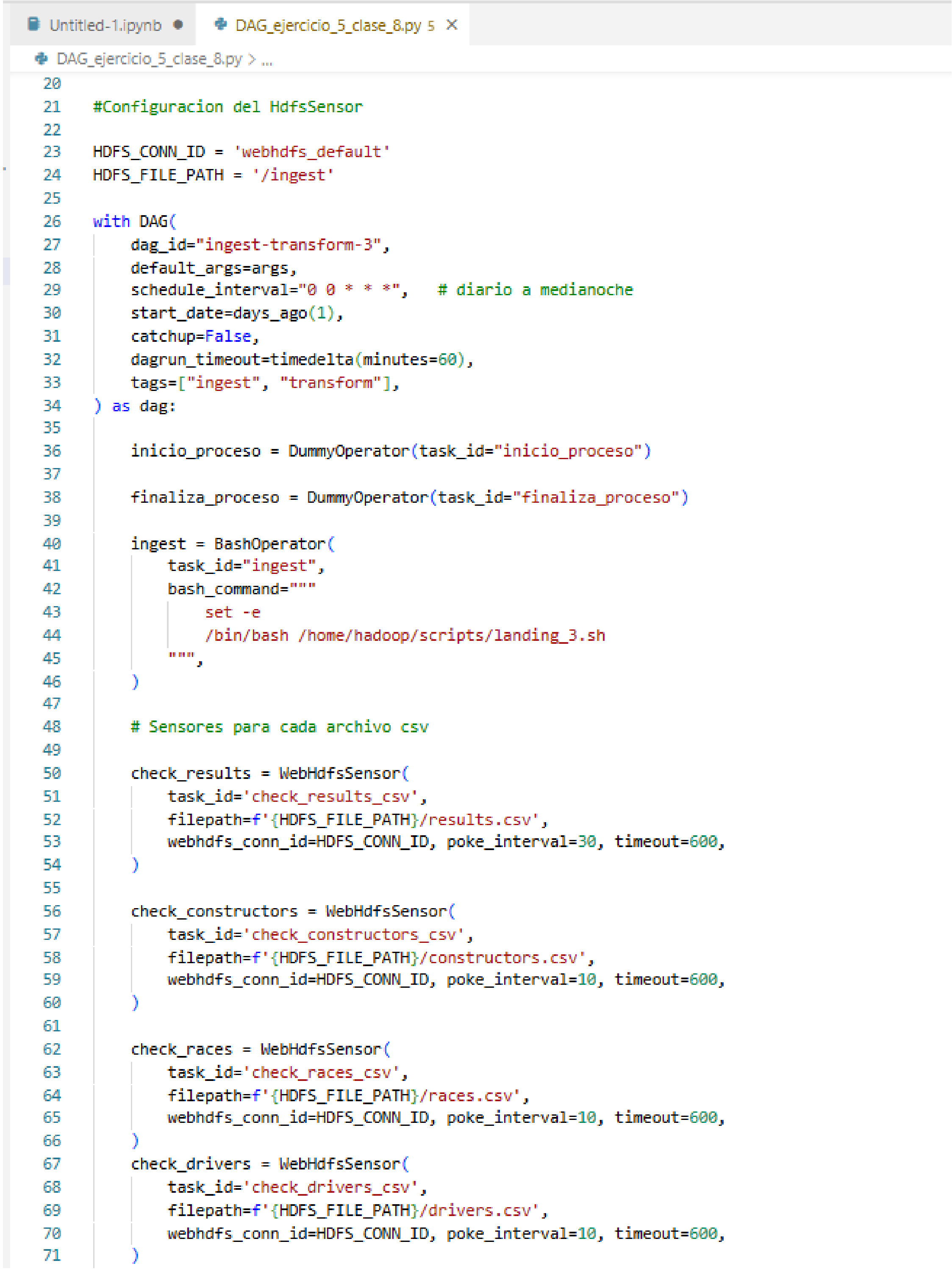
transform_load_3.py > ...

```
###################################################################################################
# Archivo: transform_load_3.py
#
# Enunciado ejercicio 4:
# 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
#
#
# Tarea: Transformar archivos csv armar dfs y cargar en Hive (f1.driver_results y f1.constructor_result)
# Requisitos:
#   - La base de datos y tablas Hive existen: f1.driver_results y f1.constructor_result
#   - Esquema de la tabla driver_results:
# driver_forename         string
# driver_surname          string
# driver_nationality      string
# points                  int
#   - Esquema de la tabla constructor_result
# constructorref          string
# cons_name               string
# cons_nationality        string
# url                     string
# points                  int
###################################################################################################

from pyspark.sql import SparkSession

# Crear sesion en Spark

def main():
    # 1) Crear la sesión de Spark con soporte Hive
    spark = (
        SparkSession
            .builder
            .appName("transform_load_3")
            .enableHiveSupport()
            .getOrCreate()
    )

# 2) Rutas de entrada (csv en HDFS)
p1 = "hdfs://172.17.0.2:9000/ingest/results.csv"
p2 = "hdfs://172.17.0.2:9000/ingest/drivers.csv"
p3 = "hdfs://172.17.0.2:9000/ingest/constructors.csv"
p4 = "hdfs://172.17.0.2:9000/ingest/races.csv"

results = spark.read.option("header", "true").format("csv").load(p1)
drivers = spark.read.option("header", "true").format("csv").load(p2)
constructors = spark.read.option("header", "true").format("csv").load(p3)
races = spark.read.option("header", "true").format("csv").load(p4)

# 3) Vistas temporales

results.createOrReplaceTempView("v_results")
drivers.createOrReplaceTempView("v_drivers")
constructors.createOrReplaceTempView("v_constructors")
races.createOrReplaceTempView("v_races")
```

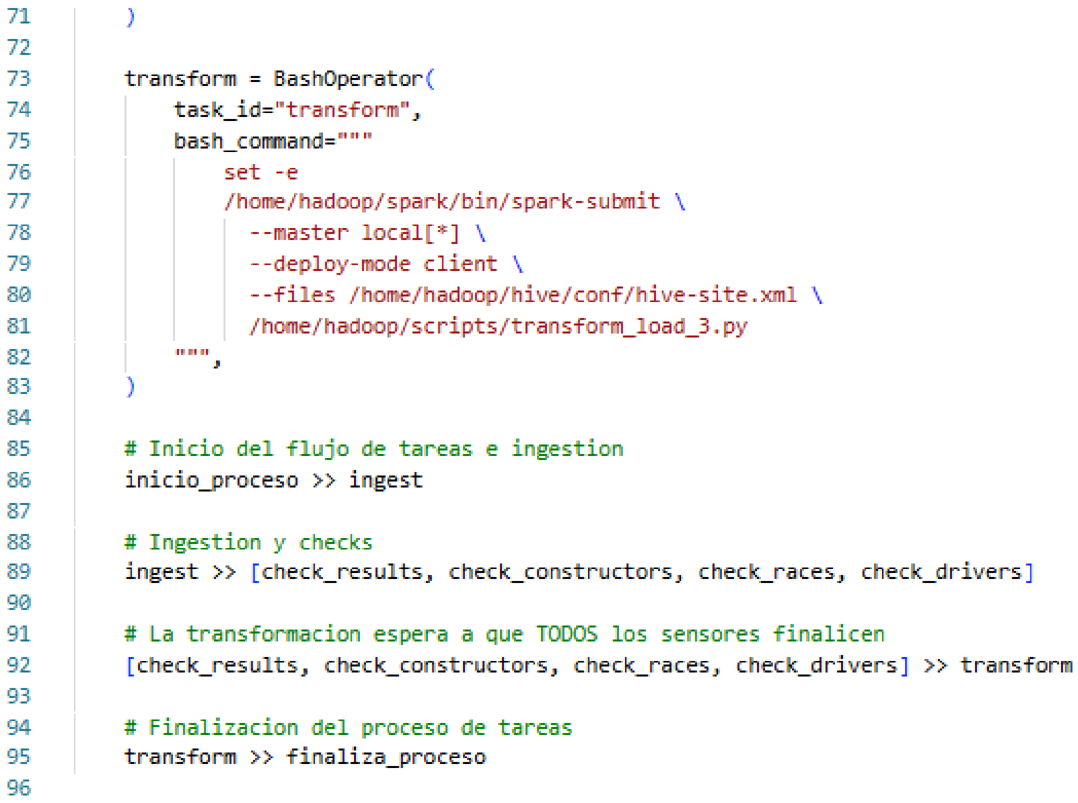


```python
# 4) Transformaciones

# Join de tablas drivers + results (incluye casteo)

df_drivers_result = spark.sql("""
SELECT
  CAST(d.forename AS STRING) AS driver_forename,
  CAST(d.surname  AS STRING) AS driver_surname,
  CAST(d.nationality AS STRING) AS driver_nationality,
  SUM(CAST(r.points AS INT))        AS points
FROM v_results r
JOIN v_drivers d
  ON r.driverId = d.driverId
GROUP BY
  CAST(d.forename AS STRING),
  CAST(d.surname  AS STRING),
  CAST(d.nationality AS STRING)
SORT BY points DESC
""")

# Join de tablas constructors + results (incluye casteo)

df_constructors_result = spark.sql("""
SELECT
  CAST(c.constructorRef AS STRING)   AS constructorref,
  CAST(c.name AS STRING)             AS cons_name,
  CAST(c.nationality AS STRING)      AS cons_nationality,
  CAST(c.url AS STRING)              AS url,
  SUM(CAST(r.points AS DOUBLE))      AS points
FROM v_results r
JOIN v_constructors c
  ON CAST(r.constructorId AS INT) = CAST(c.constructorId AS INT)
JOIN v_races ra
  ON CAST(r.raceId AS INT) = CAST(ra.raceId AS INT)
WHERE ra.circuitId = 4
  AND ra.year = 1991
GROUP BY
  CAST(c.constructorRef AS STRING),
  CAST(c.name AS STRING),
  CAST(c.nationality AS STRING),
  CAST(c.url AS STRING)
ORDER BY points DESC
""")

# 5) Loads

df_drivers_result.write.mode("append").insertInto("f1.driver_results")
df_constructors_result.write.mode("append").insertInto("f1.constructor_results")

spark.stop()

if __name__ == "__main__":
    main()
```

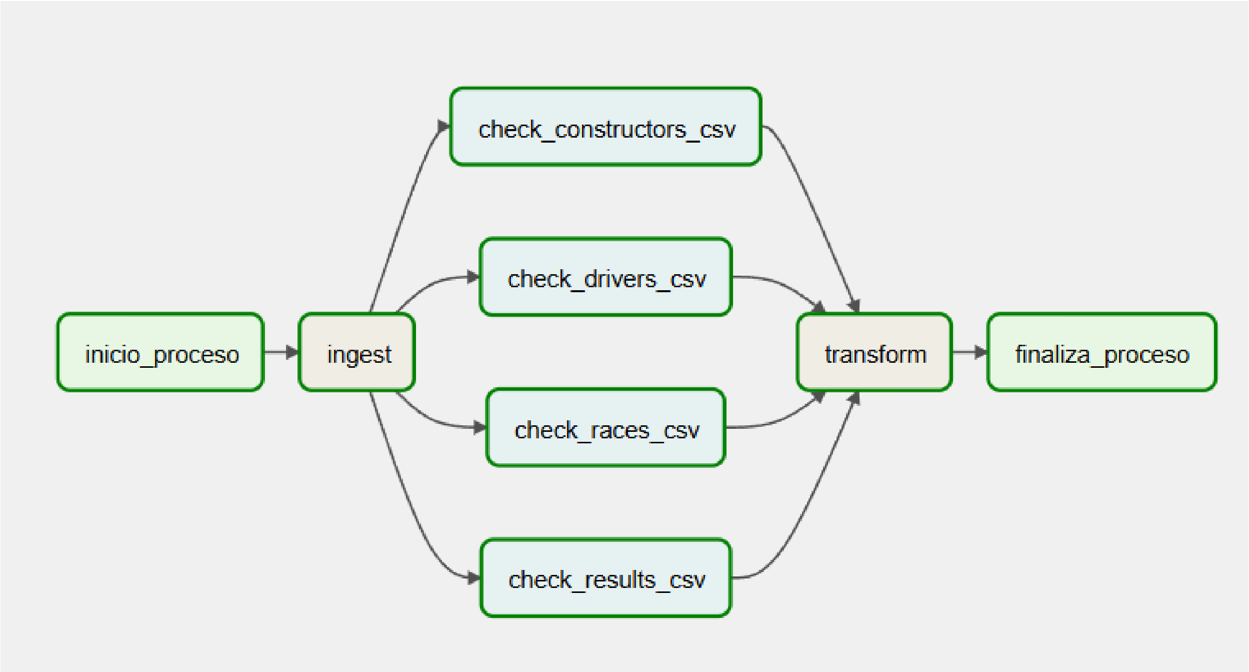
5. Realizar un proceso automático en Airflow que orqueste los archivos creados en los puntos 3 y 4. Correrlo y mostrar una captura de pantalla (del DAG y del resultado en la base de datos)


Untitled-1.ipynb ●
DAG_ejercicio_5_clase_8.py 5 ×
DAG_ejercicio_5_clase_8.py > ...

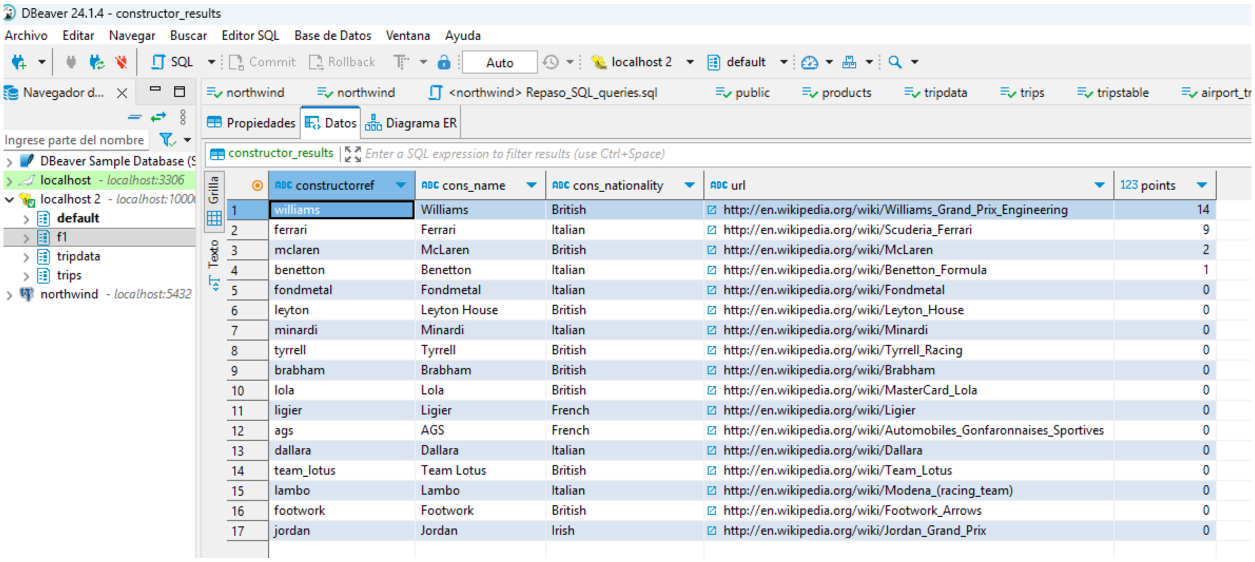
```python

#Configuracion del HdfsSensor

HDFS_CONN_ID = 'webhdfs_default'
HDFS_FILE_PATH = '/ingest'

with DAG(
    dag_id="ingest-transform-3",
    default_args=args,
    schedule_interval="0 0 * * *",    # diario a medianoche
    start_date=days_ago(1),
    catchup=False,
    dagrun_timeout=timedelta(minutes=60),
    tags=["ingest", "transform"],
) as dag:

    inicio_proceso = DummyOperator(task_id="inicio_proceso")

    finaliza_proceso = DummyOperator(task_id="finaliza_proceso")

    ingest = BashOperator(
        task_id="ingest",
        bash_command="""
            set -e
            /bin/bash /home/hadoop/scripts/landing_3.sh
        """,
    )

    # Sensores para cada archivo csv

    check_results = WebHdfsSensor(
        task_id='check_results_csv',
        filepath=f'{HDFS_FILE_PATH}/results.csv',
        webhdfs_conn_id=HDFS_CONN_ID, poke_interval=30, timeout=600,
    )

    check_constructors = WebHdfsSensor(
        task_id='check_constructors_csv',
        filepath=f'{HDFS_FILE_PATH}/constructors.csv',
        webhdfs_conn_id=HDFS_CONN_ID, poke_interval=10, timeout=600,
    )

    check_races = WebHdfsSensor(
        task_id='check_races_csv',
        filepath=f'{HDFS_FILE_PATH}/races.csv',
        webhdfs_conn_id=HDFS_CONN_ID, poke_interval=10, timeout=600,
    )
    check_drivers = WebHdfsSensor(
        task_id='check_drivers_csv',
        filepath=f'{HDFS_FILE_PATH}/drivers.csv',
        webhdfs_conn_id=HDFS_CONN_ID, poke_interval=10, timeout=600,
    )
```



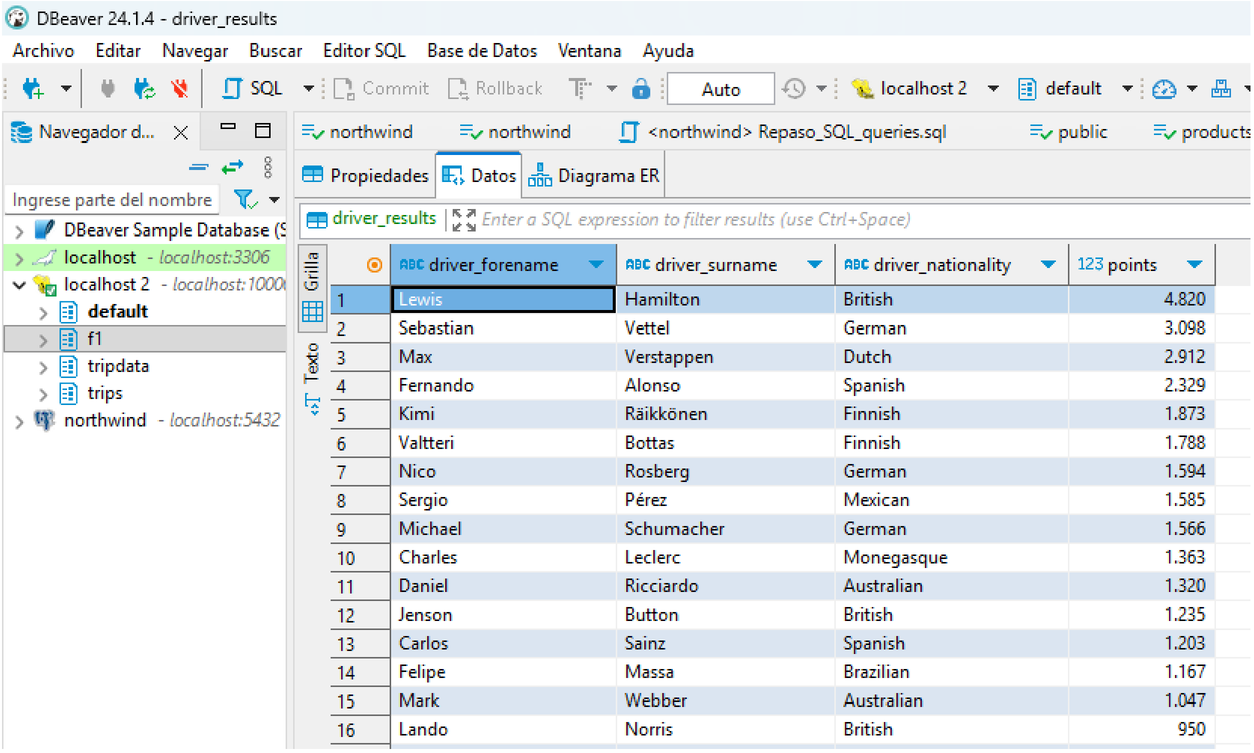
```
    )

    transform = BashOperator(
        task_id="transform",
        bash_command="""
            set -e
            /home/hadoop/spark/bin/spark-submit \
              --master local[*] \
              --deploy-mode client \
              --files /home/hadoop/hive/conf/hive-site.xml \
              /home/hadoop/scripts/transform_load_3.py
        """,
    )

    # Inicio del flujo de tareas e ingestion
    inicio_proceso >> ingest

    # Ingestion y checks
    ingest >> [check_results, check_constructors, check_races, check_drivers]

    # La transformacion espera a que TODOS los sensores finalicen
    [check_results, check_constructors, check_races, check_drivers] >> transform

    # Finalizacion del proceso de tareas
    transform >> finaliza_proceso

```



En DBeaver se puede observar el contenido de las tablas guardadas en la base f1 en Hive:



	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
5	fondmetal	Fondmetal	Italian	http://en.wikipedia.org/wiki/Fondmetal	0
6	leyton	Leyton House	British	http://en.wikipedia.org/wiki/Leyton_House	0
7	minardi	Minardi	Italian	http://en.wikipedia.org/wiki/Minardi	0
8	tyrrell	Tyrrell	British	http://en.wikipedia.org/wiki/Tyrrell_Racing	0
9	brabham	Brabham	British	http://en.wikipedia.org/wiki/Brabham	0
10	lola	Lola	British	http://en.wikipedia.org/wiki/MasterCard_Lola	0
11	ligier	Ligier	French	http://en.wikipedia.org/wiki/Ligier	0
12	ags	AGS	French	http://en.wikipedia.org/wiki/Automobiles_Gonfaronnaises_Sportives	0
13	dallara	Dallara	Italian	http://en.wikipedia.org/wiki/Dallara	0
14	team_lotus	Team Lotus	British	http://en.wikipedia.org/wiki/Team_Lotus	0
15	lambo	Lambo	Italian	http://en.wikipedia.org/wiki/Modena_(racing_team)	0
16	footwork	Footwork	British	http://en.wikipedia.org/wiki/Footwork_Arrows	0
17	jordan	Jordan	Irish	http://en.wikipedia.org/wiki/Jordan_Grand_Prix	0



	driver_forename	driver_surname	driver_nationality	points
1	Lewis	Hamilton	British	4.820
2	Sebastian	Vettel	German	3.098
3	Max	Verstappen	Dutch	2.912
4	Fernando	Alonso	Spanish	2.329
5	Kimi	Räikkönen	Finnish	1.873
6	Valtteri	Bottas	Finnish	1.788
7	Nico	Rosberg	German	1.594
8	Sergio	Pérez	Mexican	1.585
9	Michael	Schumacher	German	1.566
10	Charles	Leclerc	Monegasque	1.363
11	Daniel	Ricciardo	Australian	1.320
12	Jenson	Button	British	1.235
13	Carlos	Sainz	Spanish	1.203
14	Felipe	Massa	Brazilian	1.167
15	Mark	Webber	Australian	1.047
16	Lando	Norris	British	950