Tarea clase 7

1 y 2

```
hive> use tripdata;
OK
Time taken: 0.037 seconds
hive> show tables;
OK
airport_trips
tripdata_table
Time taken: 0.036 seconds, Fetched: 2 row(s)
hive> describe airport_trips;
OK
tpep_pickup_datetime    date
airport_fee             float
payment_type            string
tolls_amount            float
total_amount            float
Time taken: 0.061 seconds, Fetched: 5 row(s)
hive>
```

3

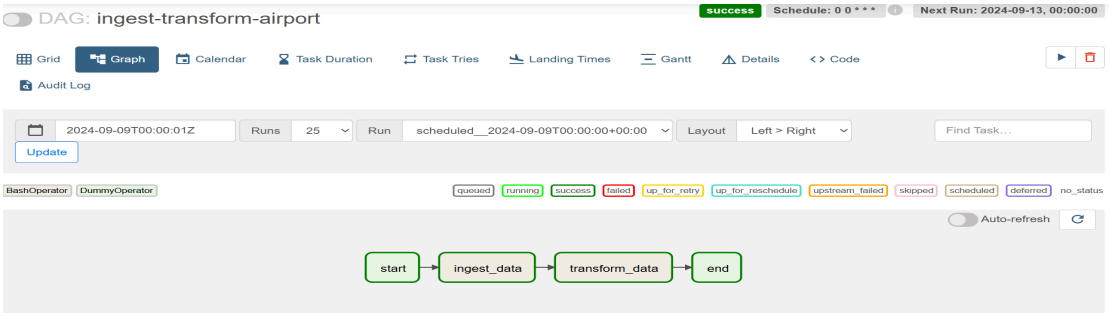
```
hadoop@f769ed737105:~/scripts$ cat ingest2.sh
TEMP_DIR="/tmp/yellow_tripdata"

mkdir -p $TEMP_DIR

URL_01="https://edvaibucket.blob.core.windows.net/data-engineer-edvai/yellow_tripdata_2021-01.parque
t?sp=r&st=2023-11-06T12:52:39Z&se=2025-11-06T20:52:39Z&sv=2022-11-02&sr=c&sig=J4Ddi2c7Ep23OhQLPisbYa
erlH472iigPwc1%2FkG80EM%3D"
URL_02="https://edvaibucket.blob.core.windows.net/data-engineer-edvai/yellow_tripdata_2021-02.parque
t?sp=r&st=2023-11-06T12:52:39Z&se=2025-11-06T20:52:39Z&sv=2022-11-02&sr=c&sig=J4Ddi2c7Ep23OhQLPisbYa
erlH472iigPwc1%2FkG80EM%3D"

FILE_01="yellow_tripdata_2021-01.parquet"
FILE_02="yellow_tripdata_2021-02.parquet"

wget -O $TEMP_DIR/$FILE_01 "$URL_01"
wget -O $TEMP_DIR/$FILE_02 "$URL_02"

hdfs dfs -put $TEMP_DIR/$FILE_01 /ingest
hdfs dfs -put $TEMP_DIR/$FILE_02 /ingest

rm -r $TEMP_DIR

echo "Archivos descargados e ingresados en HDFS con exito."

hadoop@f769ed737105:~/scripts$
```

4

```
hadoop@f769ed737105:~/scripts$ cat trans2.py



from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
sc = SparkContext('local')
spark = SparkSession(sc)
from pyspark.sql import HiveContext
hc = HiveContext(sc)


df_january = spark.read.parquet("hdfs://172.17.0.2:9000/ingest/yellow_tripdata_2021-01.parquet")
df_february = spark.read.parquet("hdfs://172.17.0.2:9000/ingest/yellow_tripdata_2021-02.parquet")

df_combined = df_january.union(df_february)

df_combined.createOrReplaceTempView("tripdata_vista")

df_castandfilt = spark.sql("""     SELECT         CAST(tpep_pickup_datetime AS date) AS tpep_pickup_
datetime,         CAST(airport_fee AS float) AS airport_fee,         CAST(payment_type AS string) AS
 payment_type,         CAST(tolls_amount AS float) AS tolls_amount,         CAST(total_amount AS flo
at) AS total_amount      FROM         tripdata_vista      WHERE          PULocationID IN (1,132,138)
 OR DOLocationID IN (1,132,138)         AND payment_type = 2 """)


df_castandfilt.createOrReplaceTempView("tripdata_castandfilt")

hc.sql("insert into tripdata.airport_trips select * from tripdata_castandfilt;")

hadoop@f769ed737105:~/scripts$
```

5

```
args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}

with DAG(
    dag_id='ingest-transform-airport',
    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:

    start_task = DummyOperator(
        task_id='start',
)

    ingest_task = BashOperator(
        task_id='ingest_data',
        bash_command='/usr/bin/sh /home/hadoop/scripts/ingest2.sh ',
)
    transform_task = BashOperator(
        task_id='transform_data',
        bash_command='ssh hadoop@172.17.0.2 /home/hadoop/spark/bin/spark-submit --files /home/hadoo
p/hive/conf/hive-site.xml /home/hadoop/scripts/trans2.py ',
)
    end_task = DummyOperator(
        task_id='end',
)
```



Tarea clase 8

1 y 2

```
hive> use f1;
OK
Time taken: 0.036 seconds
hive> show tables;
OK
constructor_results
driver_results
Time taken: 0.03 seconds, Fetched: 2 row(s)
hive> describe constructor_results;
OK
constructorref          string
cons_name               string
cons_nationality        string
url                     string
points                  float
Time taken: 0.047 seconds, Fetched: 5 row(s)
hive> describe driver_results;
OK
driver_forename         string
driver_surname          string
driver_nationality      string
points                  float
Time taken: 0.054 seconds, Fetched: 4 row(s)
hive>
```

3

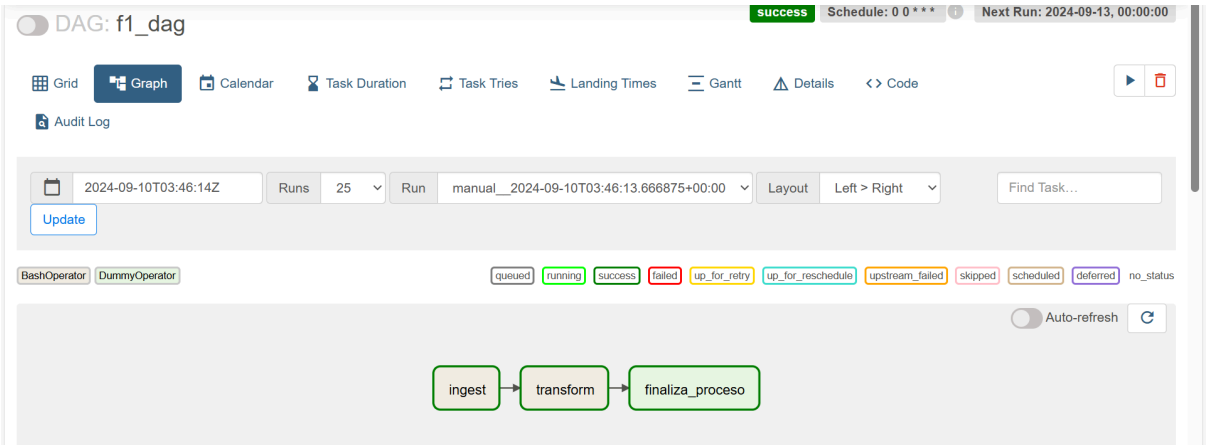
```
hadoop@f769ed737105:~/scripts$ cat ingest3.sh
TEMP_DIR="/tmp/f1"
mkdir -p $TEMP_DIR

URL_01="https://dataengineerpublic.blob.core.windows.net/data-engineer/f1/results.csv"
URL_02="https://dataengineerpublic.blob.core.windows.net/data-engineer/f1/drivers.csv"
URL_03="https://dataengineerpublic.blob.core.windows.net/data-engineer/f1/constructors.csv"
URL_04="https://dataengineerpublic.blob.core.windows.net/data-engineer/f1/races.csv"


FILE_01="results.csv"
FILE_02="drivers.csv"
FILE_03="constructors.csv"
FILE_04="races.csv"

wget -O $TEMP_DIR/$FILE_01 "$URL_01"
wget -O $TEMP_DIR/$FILE_02 "$URL_02"
wget -O $TEMP_DIR/$FILE_03 "$URL_03"
wget -O $TEMP_DIR/$FILE_04 "$URL_04"


hdfs dfs -put $TEMP_DIR/$FILE_01 /ingest
hdfs dfs -put $TEMP_DIR/$FILE_02 /ingest
hdfs dfs -put $TEMP_DIR/$FILE_03 /ingest
hdfs dfs -put $TEMP_DIR/$FILE_04 /ingest

rm -r $TEMP_DIR
echo "Archivos descargados e ingresados en HDFS con éxito."
```

4

```
hadoop@f769ed737105:~/scripts$ cat trans3.py
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("F1 Data Processing") \
    .enableHiveSupport() \
    .getOrCreate()


constructors = spark.read.option("header", "true").csv("hdfs://172.17.0.2:9000/ingest/constructors.
csv")
races = spark.read.option("header", "true").csv("hdfs://172.17.0.2:9000/ingest/races.csv")
drivers = spark.read.option("header", "true").csv("hdfs://172.17.0.2:9000/ingest/drivers.csv")
results = spark.read.option("header", "true").csv("hdfs://172.17.0.2:9000/ingest/results.csv")


constructors.createOrReplaceTempView("constructors_vista")
races.createOrReplaceTempView("races_vista")
drivers.createOrReplaceTempView("drivers_vista")
results.createOrReplaceTempView("results_vista")


driver_join = spark.sql("""
      SELECT
          CAST(drivers_vista.forename AS STRING) AS driver_forename,
          CAST(drivers_vista.surname AS STRING) AS driver_surname,
          CAST(drivers_vista.nationality AS STRING) AS driver_nationality,
          CAST(results_vista.points AS FLOAT) AS points
      FROM drivers_vista
      INNER JOIN results_vista ON drivers_vista.driverID = results_vista.driverID
      ORDER BY results_vista.points DESC
      LIMIT 10 """)


constructor_join = spark.sql("""
    SELECT
        CAST(constructors_vista.constructorRef AS STRING) AS constructorref,
        CAST(constructors_vista.name AS STRING) AS cons_name,
          CAST(constructors_vista.nationality AS STRING) AS cons_nationality,
          CAST(constructors_vista.url AS STRING) AS url,
          CAST(results_vista.points AS FLOAT) AS points
      FROM constructors_vista
      INNER JOIN results_vista ON constructors_vista.constructorId = results_vista.constructorId
      INNER JOIN races_vista ON results_vista.raceId = races_vista.raceId
      WHERE races_vista.name = 'Spanish Grand Prix' AND races_vista.year = 1991
      ORDER BY results_vista.points DESC
      LIMIT 10 """)

  driver_join.createOrReplaceTempView("driver_final")
  constructor_join.createOrReplaceTempView("constructor_final")

  spark.sql("INSERT INTO f1.driver_results SELECT * FROM driver_final;")
  spark.sql("INSERT INTO f1.constructor_results SELECT * FROM constructor_final;")


  hadoop@f769ed737105:~/scripts$
```

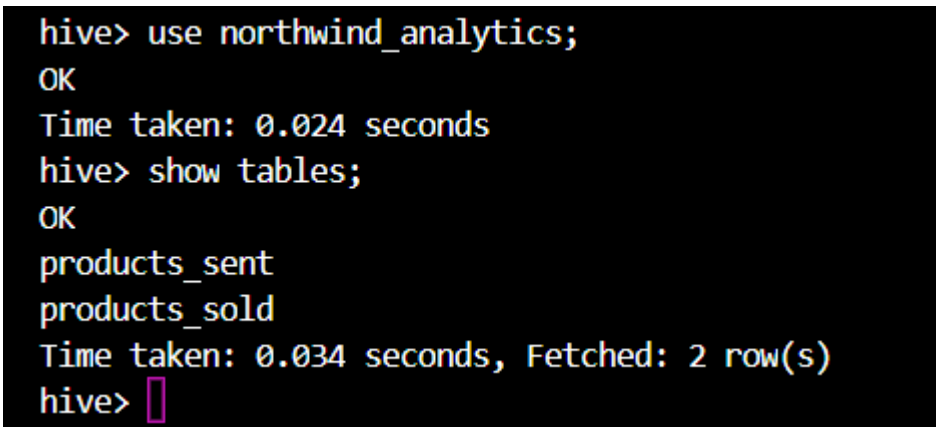
5

```
hadoop@f769ed737105:~/airflow/dags$ cat f1_dag.py
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_dag',
    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/ingest3.sh ',
)
     transform = BashOperator(
         task_id='transform',
         bash_command='ssh hadoop@172.17.0.2 /home/hadoop/spark/bin/spark-submit --files /home/hado
op/hive/conf/hive-site.xml /home/hadoop/scripts/trans3.py ',
 )

ingest >> transform >>finaliza_proceso
if __name__ == "__main__":
    dag.cli()
```



Tarea clase 9

1



2

```
hadoop@f769ed737105:~/scripts$ cat northwind_clientes.sh
DB_URL="jdbc:postgresql://172.17.0.4:5432/northwind"
DB_USERNAME="postgres"
PASSWORD="edvai"
TEMP_DIR="hdfs://172.17.0.2:9000/sqoop/ingest/temp_clientes"
FINAL_DIR="hdfs://172.17.0.2:9000/sqoop/ingest/clientes"
PERMISSIONS="777"  # Read, write, and execute permissions for all

QUERY="SELECT c.customer_id, c.company_name, SUM(od.quantity) AS productos_vendidos
       FROM customers c
       JOIN orders o ON c.customer_id = o.customer_id
       JOIN order_details od ON o.order_id = od.order_id
       GROUP BY c.customer_id, c.company_name
       HAVING \$CONDITIONS
       ORDER BY productos_vendidos DESC"

NUM_MAPPERS=1

# Run the Sqoop import
sqoop import \
  --connect $DB_URL \
  --username $DB_USERNAME \
  --password $PASSWORD \
  --query "$QUERY" \
  --target-dir $TEMP_DIR \
  --as-parquetfile \
  --delete-target-dir \
  --num-mappers $NUM_MAPPERS

if [ $? -eq 0 ]; then
  # Create the final directory if it does not exist
  hadoop fs -mkdir -p $FINAL_DIR

  # Find any .parquet file in the temporary directory
  PARQUET_FILE=$(hadoop fs -ls $TEMP_DIR | grep '.parquet' | awk '{print $8}')

  if [ -n "$PARQUET_FILE" ]; then
    # Rename the found .parquet file
    hadoop fs -mv $PARQUET_FILE $FINAL_DIR/northwind_clientes.parquet

    # Set the permissions of the final file
    hadoop fs -chmod $PERMISSIONS $FINAL_DIR/northwind_clientes.parquet
    echo "File renamed to northwind_clientes.parquet and permissions set to $PERMISSIONS."
  else
    echo "No .parquet file found in the temporary directory."
  fi

  # Remove the temporary directory
  hadoop fs -rm -r $TEMP_DIR

  echo "Client import completed successfully, file renamed to northwind_clientes, permissions set, a
nd temporary directory removed."
else
  echo "Error during client import."
fi
```

3

```
hadoop@f769ed737105:~/scripts$ cat northwind_envios.sh
#!/bin/bash

# Configuration variables
DB_URL="jdbc:postgresql://172.17.0.4:5432/northwind"
DB_USERNAME="postgres"
PASSWORD="edvai"
TEMP_DIR="hdfs://172.17.0.2:9000/sqoop/ingest/temp_envios"
FINAL_DIR="hdfs://172.17.0.2:9000/sqoop/ingest/envios"
PERMISSIONS="777"  # Read, write, and execute permissions for all

QUERY="SELECT o.order_id, o.shipped_date, c.company_name, c.phone
       FROM orders o
       JOIN customers c ON o.customer_id = c.customer_id
       WHERE \$CONDITIONS"

NUM_MAPPERS=1

# Run the Sqoop import
sqoop import \
  --connect $DB_URL \
  --username $DB_USERNAME \
  --password $PASSWORD \
  --query "$QUERY" \
  --target-dir $TEMP_DIR \
  --as-parquetfile \
  --delete-target-dir \
  --num-mappers $NUM_MAPPERS

# Check if the import was successful
if [ $? -eq 0 ]; then
  # Create the final directory if it does not exist
  hadoop fs -mkdir -p $FINAL_DIR

  # Find the .parquet file in the temporary directory
  PARQUET_FILE=$(hadoop fs -ls $TEMP_DIR | grep '.parquet' | awk '{print $8}')

  if [ -n "$PARQUET_FILE" ]; then
    # Rename the found .parquet file
    hadoop fs -mv $PARQUET_FILE $FINAL_DIR/northwind_envios.parquet

    # Set the permissions of the final file
    hadoop fs -chmod $PERMISSIONS $FINAL_DIR/northwind_envios.parquet
    echo "File renamed to northwind_envios.parquet and permissions set to $PERMISSIONS."
  else
    echo "No .parquet file found in the temporary directory."
  fi

  # Remove the temporary directory
  hadoop fs -rm -r $TEMP_DIR

  echo "Import completed successfully, file renamed to northwind_envios, permissions set, and tempor
ary directory removed."
else
  echo "Error during the import."
fi
```

4

```
hadoop@f769ed737105:~/scripts$ cat northwind_order_details.sh
#!/bin/bash

# Configuration variables
DB_URL="jdbc:postgresql://172.17.0.4:5432/northwind"
DB_USERNAME="postgres"
PASSWORD="edvai"
TEMP_DIR="hdfs://172.17.0.2:9000/sqoop/ingest/temp_order_details"
FINAL_DIR="hdfs://172.17.0.2:9000/sqoop/ingest/order_details"
PERMISSIONS="777"  # Read, write, and execute permissions for all

QUERY="SELECT order_id, unit_price, quantity, discount
       FROM order_details
       WHERE \$CONDITIONS"

NUM_MAPPERS=1

# Run the Sqoop import
sqoop import \
  --connect $DB_URL \
  --username $DB_USERNAME \
  --password $PASSWORD \
  --query "$QUERY" \
  --target-dir $TEMP_DIR \
  --as-parquetfile \
  --delete-target-dir \
  --num-mappers $NUM_MAPPERS

# Check if the import was successful
if [ $? -eq 0 ]; then
  # Create the final directory if it does not exist
  hadoop fs -mkdir -p $FINAL_DIR

  # Find the .parquet file in the temporary directory
  PARQUET_FILE=$(hadoop fs -ls $TEMP_DIR | grep '.parquet' | awk '{print $8}')

  if [ -n "$PARQUET_FILE" ]; then
    # Rename the found .parquet file
    hadoop fs -mv $PARQUET_FILE $FINAL_DIR/northwind_order_details.parquet

    # Set the permissions of the final file
    hadoop fs -chmod $PERMISSIONS $FINAL_DIR/northwind_order_details.parquet
    echo "File renamed to northwind_order_details.parquet and permissions set to $PERMISSIONS."
  else
    echo "No .parquet file found in the temporary directory."
  fi

  # Remove the temporary directory
  hadoop fs -rm -r $TEMP_DIR

  echo "Order details import completed successfully, file renamed to northwind_order_details, permis
sions set, and temporary directory removed."
else
  echo "Error during the order details import."
fi
```

5

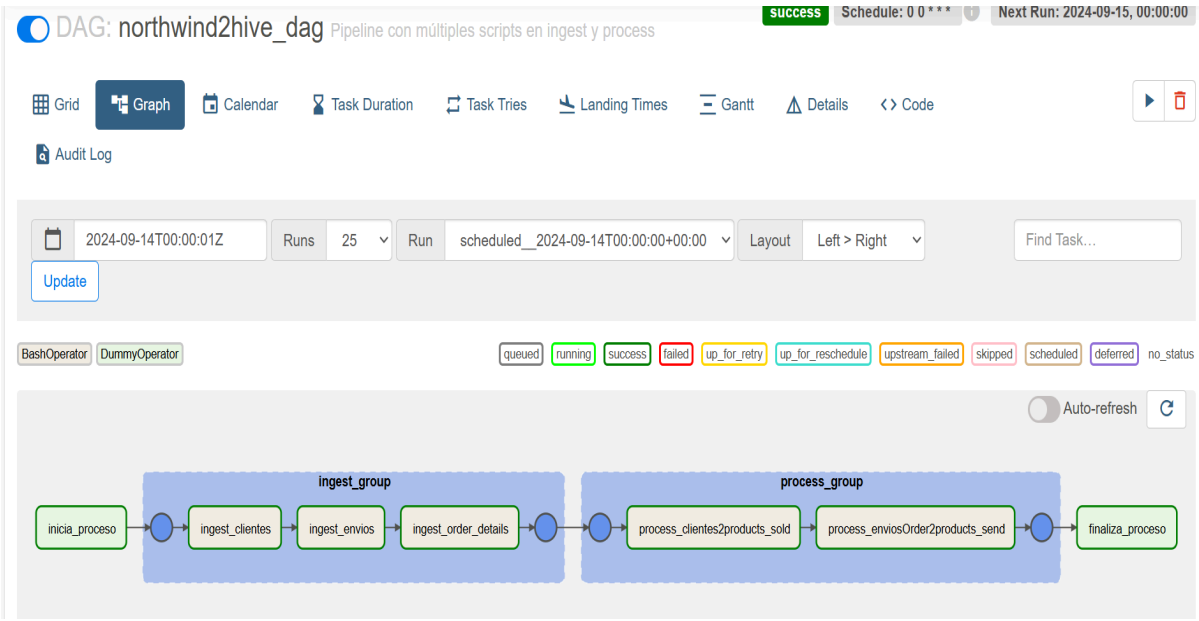
```
hadoop@f769ed737105:~/scripts$ cat clientes2products_sold.py
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
sc = SparkContext('local')
spark = SparkSession(sc)
from pyspark.sql import HiveContext
hc = HiveContext(sc)


clientes = spark.read.parquet("hdfs://172.17.0.2:9000/sqoop/ingest/clientes/northwind_clientes.parqu
et")


clientes.createOrReplaceTempView("clientes_vista")

clientes_mod = spark.sql("""
    WITH clientes_promedio AS (
        SELECT
            CAST(company_name AS string) AS company_name,
            CAST(productos_vendidos AS int) AS productos_vendidos,
            avg(CAST(productos_vendidos AS int)) OVER () AS avg_productos_vendidos
        FROM clientes_vista
    )
    SELECT *
    FROM clientes_promedio
    WHERE productos_vendidos > avg_productos_vendidos
""")




clientes_mod.createOrReplaceTempView("clientes_mod_vista")

hc.sql("insert into northwind_analytics.products_sold select * from clientes_mod_vista;")

hadoop@f769ed737105:~/scripts$
```

6

```
hadoop@f769ed737105:~/scripts$ cat enviosOrder2products_send.py
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql.functions import col, from_unixtime, expr

sc = SparkContext('local')
spark = SparkSession(sc)
from pyspark.sql import HiveContext
hc = HiveContext(sc)


envios = spark.read.parquet("hdfs://172.17.0.2:9000/sqoop/ingest/envios/northwind_envios.parquet")
order_details = spark.read.parquet("hdfs://172.17.0.2:9000/sqoop/ingest/order_details/northwind_orde
r_details.parquet")


envios.createOrReplaceTempView("envios_vista")
order_details.createOrReplaceTempView("order_details_vista")

envod_joined = spark.sql("""
select
    cast(ev.order_id as int) as order_id,
    cast(from_unixtime(ev.shipped_date / 1000, 'yyyy-MM-dd') as date) as shipped_date,  -- Convertir
 y luego hacer cast a DATE
    cast(ev.company_name as string) as company_name,
    cast(ev.phone as string) as phone,
    cast(ov.unit_price * (1 - ov.discount) as float) as unit_price_discount,  -- Aplicar descuento a
l precio unitario
    cast(ov.quantity as int) as quantity,
    cast((ov.unit_price * (1 - ov.discount) * ov.quantity) as float) as total_price  -- Calcular el
total_price con descuento
from
    envios_vista ev
inner join
    order_details_vista ov
on
    ev.order_id = ov.order_id
where
    ov.discount <> 0
 """)


 envod_joined.createOrReplaceTempView("envod_joined_vista")

 hc.sql("insert into northwind_analytics.products_sent  select * from envod_joined_vista;")
 hadoop@f769ed737105:~/scripts$
```

7

```
hadoop@f769ed737105:~/airflow/dags$ cat northwind2hive.py
from datetime import timedelta
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.operators.dummy import DummyOperator
from airflow.utils.dates import days_ago
from airflow.utils.task_group import TaskGroup


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

with DAG(
    dag_id='northwind2hive_dag',
    default_args= args,
    description='Pipeline con múltiples scripts en ingest y process',
    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',
    )

    # Grupo de tareas para Ingest
    with TaskGroup('ingest_group') as ingest_group:
        ingest_clientes = BashOperator(
            task_id='ingest_clientes',
            bash_command='/usr/bin/sh /home/hadoop/scripts/northwind_clientes.sh ',
        )
        ingest_envios = BashOperator(
            task_id='ingest_envios',
            bash_command='/usr/bin/sh /home/hadoop/scripts/northwind_envios.sh ',
        )

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

        # Definir el orden en que se ejecutan los scripts de ingestión
        ingest_clientes >> ingest_envios >> ingest_order_details

    # Grupo de tareas para Process
    with TaskGroup('process_group') as process_group:
        process_clientes2products_sold = BashOperator(
            task_id='process_clientes2products_sold',
            bash_command='ssh hadoop@172.17.0.2 /home/hadoop/spark/bin/spark-submit --files /home/ha
doop/hive/conf/hive-site.xml /home/hadoop/scripts/clientes2products_sold.py ',
        )
        process_enviosOrder2products_send = BashOperator(
            task_id='process_enviosOrder2products_send',
            bash_command='ssh hadoop@172.17.0.2 /home/hadoop/spark/bin/spark-submit --files /home/ha
doop/hive/conf/hive-site.xml /home/hadoop/scripts/enviosOrder2products_send.py ',
        )

        # Definir el orden en que se ejecutan los scripts de procesamiento
        process_clientes2products_sold >> process_enviosOrder2products_send

    finaliza_proceso = DummyOperator(
        task_id='finaliza_proceso',
    )

    # Definir la secuencia de ejecución completa
    inicia_proceso >> ingest_group >> process_group >> finaliza_proceso


if __name__ == "__main__":
    dag.cli()
hadoop@f769ed737105:~/airflow/dags$
```


DAG: northwind2hive_dag
Pipeline con múltiples scripts en ingest y process
success
Schedule: 0 0 * * *
Next Run: 2024-09-15, 00:00:00
Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Audit Log
2024-09-14T00:00:01Z
Runs
25
Run
scheduled__2024-09-14T00:00:00+00:00
Layout
Left > Right
Find Task...
Update
BashOperator
DummyOperator
queued
running
success
failed
up_for_retry
up_for_reschedule
upstream_failed
skipped
scheduled
deferred
no_status
Auto-refresh
ingest_group
process_group
inicia_proceso
ingest_clientes
ingest_envios
ingest_order_details
process_clientes2products_sold
process_enviosOrder2products_send
finaliza_proceso