1. En Hive, crear la siguiente tabla (externa) en la base de datos tripdata:
a. airport_trips(tpep_pickup_datetetime, airport_fee, payment_type, tolls_amount, total_amount)

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

```
hive> CREATE EXTERNAL TABLE tripdata.airports_trips(tpep_pickup_datetetime date, airport_fee float, payment_type int, tolls_amount double, total_amount double)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > LOCATION '/tables/external/tripdata';
OK
Time taken: 1.664 seconds
```

```
hive> show tables;
OK
airports_trips
congestion
distance
passengers
payments
tolls
tripdata_table
Time taken: 0.091 seconds, Fetched: 7 row(s)
hive>
```

2. En Hive, mostrar el esquema de airport_trips

```
hive> describe airports_trips;
OK
tpep_pickup_datetetime  date
airport_fee             float
payment_type            int
tolls_amount            double
total_amount            double
Time taken: 0.105 seconds, Fetched: 5 row(s)
hive>
```

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

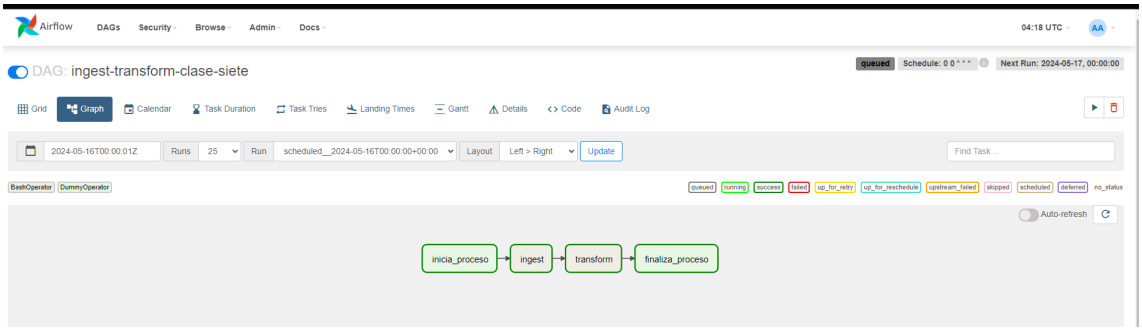
```
hadoop@d41c15beb563:~/scripts$ ls
derby.log  github.com  ingest.sh  ingest_claseSiete.sh  landing.sh  spark-warehouse  start-services.sh  transformation.py
hadoop@d41c15beb563:~/scripts$ cat ingest_claseSiete.sh
rm -f /home/hadoop/landing/*.*

wget -P /home/hadoop/landing https://dataengineerpublic.blob.core.windows.net/data-engineer/yellow_tripdata_2021-01.parquet

wget -P /home/hadoop/landing https://dataengineerpublic.blob.core.windows.net/data-engineer/yellow_tripdata_2021-02.parquet

/home/hadoop/hadoop/bin/hdfs dfs -rm /ingest/*.*

/home/hadoop/hadoop/bin/hdfs dfs -put /home/hadoop/landing/*.* /ingest
hadoop@d41c15beb563:~/scripts$
```

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.

```
hadoop@d41c15beb563:~/scripts$ cat transformation_clasesiete.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)

##leo los parquets de HDFS y lo cargo en un dataframe
df = spark.read.option("header", "true").parquet("hdfs://172.17.0.2:9000/ingest/yellow_tripdata_2021-01.parquet")
df2 = spark.read.option("header", "true").parquet("hdfs://172.17.0.2:9000/ingest/yellow_tripdata_2021-02.parquet")

##creamos una vista de los DF
df.createOrReplaceTempView("tripdata_vista01")
df2.createOrReplaceTempView("tripdata_vista02")

##unimos y filtramos el DF
df_filtro = spark.sql("select * from tripdata_vista01 union all select * from tripdata_vista02")

df_filtro.createOrReplaceTempView("tripdata_vista_union")

df_filtro = spark.sql("select * from tripdata_vista_union where month(tpep_pickup_datetime) <= 02 and year(tpep_pickup_datetime) = 2021")

df_filtro.createOrReplaceTempView("tripdata_vista_filtrado_fecha")

df_filtro = spark.sql("select * from tripdata_vista_filtrado_fecha where RatecodeId = 2 or coalesce(airport_fee, 0) >0 ")

df_filtro.createOrReplaceTempView("tripdata_vista_filtrado_aeropuerto")

df_filtro = spark.sql("select * from tripdata_vista_filtrado_aeropuerto where payment_type = 2")

##Creamos una vista con la data filtrada final###
df_filtro.createOrReplaceTempView("tripdata_vista_final")

##insertamos el DF filtrado en la tabla tripdata.airport.trips
spark.sql("insert into tripdata.airport_trips select cast(tpep_pickup_datetetime as date), cast(airport_fee as float), cast(payment_type as int), cast(tolls_amount as double), cast(total_amount as double) from tripdata_vista_final")
hadoop@d41c15beb563:~/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-clase-siete',
    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_claseSiete.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/transformation_clasesiete.py ',
    )


    inicia_proceso >> ingest >> transform >>finaliza_proceso
```



```
2021-02-28      NULL    2       6.12    61.42
Time taken: 2.545 seconds, Fetched: 5753 row(s)
hive>
```

```
hive> select * from airports_trips limit 10;
OK
2021-01-01      NULL    2       0.0     55.3
2021-01-01      NULL    2       6.12    58.92
2021-01-01      NULL    2       6.12    61.42
2021-01-01      NULL    2       6.12    58.92
2021-01-01      NULL    2       0.0     55.3
2021-01-01      NULL    2       6.12    61.42
2021-01-01      NULL    2       0.0     55.3
2021-01-01      NULL    2       0.0     55.3
2021-01-01      NULL    2       0.0     52.8
2021-01-01      NULL    2       6.12    61.42
Time taken: 0.274 seconds, Fetched: 10 row(s)
hive>
```