Clase 7

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

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: a. airport_trips(tpep_pickup_datetetime, airport_fee, payment_type, tolls_amount, total_amount)

```
create external table airport_trips(tpep_pickup_datetetime DATE, airport_fee FLOAT, payment_type STRING,
        tolls_amount FLOAT,
total_amount FLOAT)
    -- row format delimited
    -- fields terminated by ','
    -- location '/tables/external/tripdata';
```

```
hive> show tables;
OK
tripdata_table
Time taken: 0.237 seconds, Fetched: 1 row(s)
hive> create external table airport_trips(tpep_pickup_datetetime DATE, airport_fee FLOAT, payment_type STRING, tolls_amount FLOA
T,
    > total_amount FLOAT)
    >     -- row format delimited
    >     -- fields terminated by ','
    >     -- location '/tables/external/tripdata';
    > ;
OK
Time taken: 1.961 seconds
hive> describe formatted airport_trips;
```

2. En Hive, mostrar el esquema de airport_trips

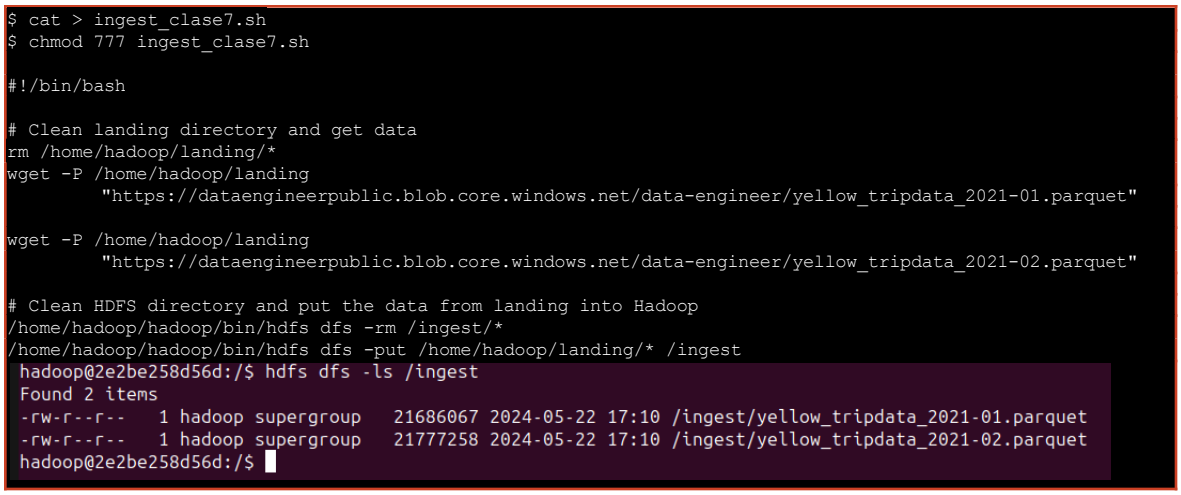
```
describe formatted airport_trips;
```

```
OK
# col_name              data_type               comment

tpep_pickup_datetetime  date
airport_fee             float
payment_type            string
tolls_amount            float
total_amount            float

# Detailed Table Information
Database:               tripdata
Owner:                  hadoop
CreateTime:             Wed May 22 18:55:28 ART 2024
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://172.17.0.2:9000/user/hive/warehouse/tripdata.db/airport_trips
```

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

   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)

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



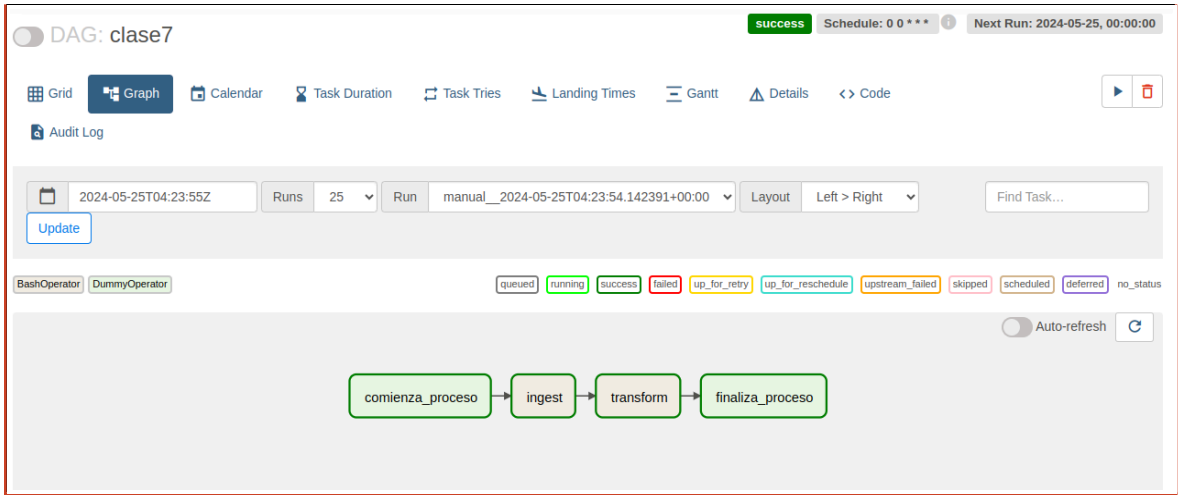
```
$ cat > ingest_clase7.sh
$ chmod 777 ingest_clase7.sh

#!/bin/bash

# Clean landing directory and get data
rm /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"

# Clean HDFS directory and put the data from landing into Hadoop
/home/hadoop/hadoop/bin/hdfs dfs -rm /ingest/*
/home/hadoop/hadoop/bin/hdfs dfs -put /home/hadoop/landing/* /ingest
hadoop@2e2be258d56d:/$ hdfs dfs -ls /ingest
Found 2 items
-rw-r--r--   1 hadoop supergroup   21686067 2024-05-22 17:10 /ingest/yellow_tripdata_2021-01.parquet
-rw-r--r--   1 hadoop supergroup   21777258 2024-05-22 17:10 /ingest/yellow_tripdata_2021-02.parquet
hadoop@2e2be258d56d:/$
```

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.

```
$ cat > spark_clase7.py
$ chmod 777 spark_clase7.py

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

# Configuramos...
sc = SparkContext('local')
spark = SparkSession(sc)
hc = HiveContext(sc)

# Traemos las tablas
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")

# Unimos las tablas
dfunion = df.union(df2)


# Creamos la vista para trabajar con sql
dfunion.createOrReplaceTempView("v_air")


# Selecciono las tablas que necesito, agrego el where y las casteo
# considero no hace falta usar dos o mas pasos para esta transformacion

df_transform = spark.sql("""
        SELECT
            CAST(tpep_pickup_datetime AS DATE),
            CAST(airport_fee AS FLOAT),
            CAST(payment_type AS INT),
            CAST(tolls_amount AS FLOAT),
            CAST(total_amount AS FLOAT)
        FROM v_air
        WHERE airport_fee IS NOT NULL AND payment_type = 2
        """)



df_transform.write.insertInto("tripdata.airport_trips")
#df_transform.createOrReplaceTempView("a_trips")
#spark.sql("insert into tripdata.airport_trips SELECT * from a_trips")

# el insert me es mas facil con python ...
```

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)

```
$ cat > airflow_clase7.py
$ chmod 777 airflow_clase7.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='clase7',
    default_args=args,
    schedule_interval='0 0 * * *',
    start_date=days_ago(2),
    dagrun_timeout=timedelta(minutes=60),
    tags=['ingest_clase7', 'transform_clase7'],
    params={"example_key": "example_value"},
) as dag:

    comienza_proceso = DummyOperator(
        task_id='comienza_proceso',
    )

    finaliza_proceso = DummyOperator(
        task_id='finaliza_proceso',
    )


    ingest = BashOperator(
        task_id='ingest',
        bash_command='/usr/bin/sh /home/hadoop/scripts/ingest_clase7.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/spark_clase7.py',
    )


    comienza_proceso >> ingest >> transform >> finaliza_proceso




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


DAG: clase7
success
Schedule: 0 0 * * *
Next Run: 2024-05-25, 00:00:00
Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Audit Log
2024-05-25T04:23:55Z
Runs
25
Run
manual__2024-05-25T04:23:54.142391+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
comienza_proceso
ingest
transform
finaliza_proceso