

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_datetime, airport_fee, payment_type, tolls_amount, total_amount)

Dentro de Hive, escribimos (la base de datos ha sido creada previamente)

```
hive> use tripdata;
OK
Time taken: 0.706 seconds
```

```
hive> create external table tripdata.airport_trips(tpep_pickup_datetime date, airport_fee float, payment_type int, tolls_amount float, total_amount float)
> row format delimited
> fields terminated by ','
> location '/tables/external/tripdata';
OK
Time taken: 0.73 seconds
hive>
```

2. En Hive, mostrar el esquema de airport_trips

```
hive> describe formatted airport_trips;
OK
# col_name          data_type          comment
tpep_pickup_datetime date
airport_fee         float
payment_type        int
tolls_amount        float
total_amount        float

# Detailed Table Information
Database:            tripdata
Owner:               hadoop
CreateTime:          Sat Oct 18 15:22:56 ART 2025
LastAccessTime:      UNKNOWN
Retention:           0
Location:             hdfs://172.17.0.2:9000/tables/external/tripdata
Table Type:          EXTERNAL_TABLE
Table Parameters:
    EXTERNAL          TRUE
    transient_lastDdlTime 1760811776

# 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: 0.087 seconds, Fetched: 31 row(s)
```

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

https://data-engineer-edvai-public.s3.amazonaws.com/yellow_tripdata_2021-01.parquet

https://data-engineer-edvai-public.s3.amazonaws.com/yellow_tripdata_2021-02.parquet

Entramos al directorio `/home/hadoop/scripts`

```
hadoop@615cf53bef6c:~$ cd /home/hadoop/scripts
hadoop@615cf53bef6c:~/scripts$ ls
Notebook  derby.log  ingest.sh  landing.sh  pyspark_jupyter.sh  spark-warehouse  start-services.sh  transformation.py
hadoop@615cf53bef6c:~/scripts$ |
```

Creamos el script **`landing_2.sh`**, que va a descargar los archivos e ingestarlos en HDFS

```
hadoop@615cf53bef6c:~/scripts$ cat > landing_2.sh
# Punto 3 (script: landing_2.sh)

# Descarga datos desde el repositorio al directorio /home/hadoop/landing
wget -P /home/hadoop/landing https://data-engineer-edvai-public.s3.amazonaws.com/yellow_tripdata_2021-01.parquet
wget -P /home/hadoop/landing https://data-engineer-edvai-public.s3.amazonaws.com/yellow_tripdata_2021-02.parquet

# Lleva el archivo a HDFS al directorio /ingest
hdfs dfs -put /home/hadoop/landing/yellow_tripdata_2021-01.parquet /ingest
hdfs dfs -put /home/hadoop/landing/yellow_tripdata_2021-02.parquet /ingest

# Borra el archivo starwars.csv del directorio /home/hadoop/landing/
rm /home/hadoop/landing/yellow_tripdata_2021-01.parquet
rm /home/hadoop/landing/yellow_tripdata_2021-02.parquet
hadoop@615cf53bef6c:~/scripts$ |
```

Consultamos permisos

```
hadoop@615cf53bef6c:~/scripts$ ls -l
total 36
drwxrwxr-x 3 hadoop hadoop 4096 Sep 29 19:46 Notebook
-rw-rw-r-- 1 hadoop hadoop 670 Feb 28 2022 derby.log
-rwxrwxr-x 1 hadoop hadoop 272 Sep 22 01:14 ingest.sh
-r-xr-xr-x 1 hadoop hadoop 398 Sep 17 16:57 landing.sh
-rw-rw-r-- 1 hadoop hadoop 716 Oct 18 16:24 landing_2.sh
-rwxrwxrwx 1 hadoop hadoop 273 Sep 28 12:14 pyspark_jupyter.sh
drwxr-xr-x 2 hadoop hadoop 4096 Feb 9 2022 spark-warehouse
-rwxrwxrwx 1 hadoop hadoop 1089 May 9 2022 start-services.sh
-rwxrwxrwx 1 hadoop hadoop 1058 May 9 2022 transformation.py
hadoop@615cf53bef6c:~/scripts$ |
```

Asignamos permisos de ejecución a **`landing_2.sh`** y verificamos que se pueda ejecutar

```
hadoop@615cf53bef6c:~/scripts$ chmod 777 landing_2.sh
hadoop@615cf53bef6c:~/scripts$ ls -l
total 36
drwxrwxr-x 3 hadoop hadoop 4096 Sep 29 19:46 Notebook
-rw-rw-r-- 1 hadoop hadoop 670 Feb 28 2022 derby.log
-rwxrwxr-x 1 hadoop hadoop 272 Sep 22 01:14 ingest.sh
-r-xr-xr-x 1 hadoop hadoop 398 Sep 17 16:57 landing.sh
-rwxrwxrwx 1 hadoop hadoop 716 Oct 18 16:24 landing_2.sh
-rwxrwxrwx 1 hadoop hadoop 273 Sep 28 12:14 pyspark_jupyter.sh
drwxr-xr-x 2 hadoop hadoop 4096 Feb 9 2022 spark-warehouse
-rwxrwxrwx 1 hadoop hadoop 1089 May 9 2022 start-services.sh
-rwxrwxrwx 1 hadoop hadoop 1058 May 9 2022 transformation.py
hadoop@615cf53bef6c:~/scripts$ |
```

Ejecutamos el archivo *landing_2.sh*

```
hadoop@615cf53bef6c:~/scripts$ ./landing_2.sh
--2025-10-18 16:32:59-- https://data-engineer-edvai-public.s3.amazonaws.com/yellow_tripdata_2021-01.parquet
Resolving data-engineer-edvai-public.s3.amazonaws.com (data-engineer-edvai-public.s3.amazonaws.com)... 16.182.40.105, 54.231.160.17, 16.182.72.57, ...
Connecting to data-engineer-edvai-public.s3.amazonaws.com (data-engineer-edvai-public.s3.amazonaws.com)[16.182.40.105]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 21686067 (21M) [application/vnd.apache.parquet]
Saving to: '/home/hadoop/landing/yellow_tripdata_2021-01.parquet'

yellow_tripdata_2021-01.parquet 100%[=====] 20.68M 449KB/s in 60s

2025-10-18 16:34:00 (353 KB/s) - '/home/hadoop/landing/yellow_tripdata_2021-01.parquet' saved [21686067/21686067]

--2025-10-18 16:34:00-- https://data-engineer-edvai-public.s3.amazonaws.com/yellow_tripdata_2021-02.parquet
Resolving data-engineer-edvai-public.s3.amazonaws.com (data-engineer-edvai-public.s3.amazonaws.com)... 16.182.68.161, 52.217.234.129, 3.5.20.102, ...
Connecting to data-engineer-edvai-public.s3.amazonaws.com (data-engineer-edvai-public.s3.amazonaws.com)[16.182.68.161]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 21777258 (21M) [application/vnd.apache.parquet]
Saving to: '/home/hadoop/landing/yellow_tripdata_2021-02.parquet'

yellow_tripdata_2021-02.parquet 100%[=====] 20.77M 317KB/s in 48s

2025-10-18 16:34:50 (439 KB/s) - '/home/hadoop/landing/yellow_tripdata_2021-02.parquet' saved [21777258/21777258]

hadoop@615cf53bef6c:~/scripts$ |
```

Verificamos que realmente se borraron los archivos de */home/hadoop/landing* y que fueron correctamente ingresados en HDFS

```
hadoop@615cf53bef6c:~$ ls /home/hadoop/landing
starwars.csv 'yellow-taxi-ny?resource=download' yellow_tripdata_2021-01.csv

hadoop@615cf53bef6c:~$ hdfs dfs -ls /ingest
Found 4 items
-rw-r--r-- 1 hadoop supergroup 5462 2025-09-26 18:19 /ingest/starwars.csv
-rw-r--r-- 1 hadoop supergroup 125981363 2025-09-28 12:55 /ingest/yellow_tripdata_2021-01.csv
-rw-r--r-- 1 hadoop supergroup 21686067 2025-10-18 16:34 /ingest/yellow_tripdata_2021-01.parquet
-rw-r--r-- 1 hadoop supergroup 21777258 2025-10-18 16:34 /ingest/yellow_tripdata_2021-02.parquet
hadoop@615cf53bef6c:~$ |
```

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.

```
#####  
#####
```

```
# Archivo: transform_load_2.py
```

```
# Tarea: Transformar Yellow Taxi (2021-01/02) y cargar en Hive (tripdata.airport_trips)
```

```
# Requisitos:
```

```
# - La base de datos y tabla Hive existen: tripdata.airport_trips
```

```
# - Esquema de la tabla:
```

```
# tpep_pickup_datetime DATE
```

```
# airport_fee      FLOAT
```

```
# payment_type     INT
```

```
# tolls_amount     FLOAT
```

```
# total_amount     FLOAT
```

```
#####  
#####
```

```
from pyspark.sql import SparkSession
```

```
from pyspark.sql.functions import col, coalesce, lit
```

```
def main():
```

```
    # 1) Crear la sesión de Spark con soporte Hive
```

```
    spark = (
```

```
        SparkSession.builder
```

```
            .appName("transform_load")
```

```
            .enableHiveSupport()
```

```
# .config("hive.metastore.uris", "thrift://metastore:9083") # si usas metastore remoto
# .config("spark.sql.warehouse.dir", "/user/hive/warehouse") # ajusta si aplica
.getOrCreate()
)
```

2) Rutas de entrada (Parquet en HDFS)

```
p1 = "hdfs://172.17.0.2:9000/ingest/yellow_tripdata_2021-01.parquet"
```

```
p2 = "hdfs://172.17.0.2:9000/ingest/yellow_tripdata_2021-02.parquet"
```

```
df1 = spark.read.parquet(p1)
```

```
df2 = spark.read.parquet(p2)
```

3) Filtro de viajes de aeropuerto:

```
# - airport_fee > 0 (con COALESCE para evitar null)
```

```
# - PULocationID o DOLocationID en {132 (JFK), 138 (LGA)}
```

```
# - RatecodeID en {2, 3} (tarifas típicas asociadas a aeropuertos)
```

```
def airport_filter(df):
```

```
    return df.where(
```

```
        (coalesce(col("airport_fee"), lit(0.0)) > 0.0) |
```

```
        (col("PULocationID").isin(132, 138)) |
```

```
        (col("DOLocationID").isin(132, 138)) |
```

```
        (col("RatecodeID").isin(2, 3))
```

```
    )
```

```
df1_f = airport_filter(df1)
```

```
df2_f = airport_filter(df2)
```

4) Unir meses asegurando el match por nombre de columnas

```
df_union = df1_f.unionByName(df2_f)
```

5) Filtrar sólo pagos en efectivo (payment_type = 2)

```
df_cash = df_union.where(col("payment_type") == 2)
```

6) Seleccionar y castear columnas EXACTAMENTE como la tabla Hive destino

```
df_load = df_cash.selectExpr(  
    "CAST(tpep_pickup_datetime AS DATE) AS tpep_pickup_datetime",  
    "CAST(airport_fee AS FLOAT) AS airport_fee",  
    "CAST(payment_type AS INT) AS payment_type",  
    "CAST(tolls_amount AS FLOAT) AS tolls_amount",  
    "CAST(total_amount AS FLOAT) AS total_amount"  
)
```

7) Insertar en tabla externa (append). Requiere que exista tripdata.airport_trips

```
df_load.write.mode("append").insertInto("tripdata.airport_trips")
```

(Opcional) Log rápido

```
inserted = df_load.count()
```

```
print(f"[OK] Filas insertadas en tripdata.airport_trips: {inserted}")
```

```
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)

Creemos el DAG dentro del directorio donde se alojan en Airflow: */home/hadoop/airflow/dags/*

```
hadoop@615cf53bef6c:~/airflow/dags$ cat > DAG_ejercicio_5.py
```

```
from datetime import timedelta
```

```
from airflow import DAG
```

```
from airflow.operators.bash import BashOperator
```

```
from airflow.operators.dummy import DummyOperator # en 2.8+ preferir EmptyOperator
```

```
from airflow.utils.dates import days_ago
```

```
args = {
```

```
    "owner": "airflow",
```

```
    "retries": 0,
```

```
}
```

```
with DAG(
```

```
    dag_id="ingest-transform-2",
```

```
    default_args=args,
```

```
    schedule_interval="0 0 * * *", # diario a medianoche
```

```
    start_date=days_ago(2),
```

```
    catchup=False,
```

```
    dagrun_timeout=timedelta(minutes=60),
```

```
    max_active_runs=1,
```

```
    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_2.sh
    """
)

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_2.py
    """
    ,
    # Si necesitas pasar conf/vars explícitas, descomenta:
    # env={
    #     "HADOOP_CONF_DIR": "/home/hadoop/hadoop/etc/hadoop",
    #     "JAVA_HOME": "/usr/lib/jvm/java-8-openjdk-amd64",
    #     "SPARK_HOME": "/home/hadoop/spark",
    #     "PATH": "/home/hadoop/spark/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
    # }
)

inicio_proceso >> ingest >> transform >> finaliza_proceso

```

Verificamos que el DAG se creo correctamente en el directorio especificado:

```

hadoop@615cf53bef6c:~/airflow/dags$ ls
DAG_ejercicio_5.py  __pycache__  example-DAG.py  ingest-transform.py

```

Comprobamos que tenga permisos


```
hadoop@615cf53bef6c:~/airflow/dags$ ls -l
total 16
-rw-rw-r-- 1 hadoop hadoop 1142 Oct 18 21:21 DAG_ejercicio_5.py
drwxrwxr-x 1 hadoop hadoop 4096 May 5 2022 __pycache__
-rw-rw-r-- 1 hadoop hadoop 1079 May 1 2022 example-DAG.py
-rw-rw-r-- 1 hadoop hadoop 1024 May 5 2022 ingest-transform.py
```

Reiniciamos Airflow

```
hadop@615cf53bef6c:~/airflow/dags$ pkill -f "airflow scheduler"
era unos segundos (2-3) y luego reinicia:
airflow scheduler -D
airflow webserver -p 8010 -D
hadop@615cf53bef6c:~/airflow/dags$ pkill -f "airflow webserver"
hadop@615cf53bef6c:~/airflow/dags$ # Espera unos segundos (2-3) y luego reinicia:
hadop@615cf53bef6c:~/airflow/dags$ airflow scheduler -D
```



```
-----  
--- | _(_) -----  
-- /| | / -- / -- / -- / \ | // ---  
-/ / | / / / / / / / / / \ | // ---  
hadop@615cf53bef6c:~/airflow/dags$ airflow webserver -p 8010 -D  
-----  
--- | _(_) -----  
-- /| | / -- / -- / -- / \ | // ---  
-/ / | / / / / / / / / / \ | // ---  
Running the Gunicorn Server with:  
Workers: 4 sync  
Host: 0.0.0.0:8010  
Timeout: 120  
Logfiles: - -  
Access Logformat:
```



```
[2025-10-18 21:30:21,597] {dagbag.py:507} INFO - Filling up the DagBag from /dev/null  
[2025-10-18 21:30:21,751] {manager.py:585} INFO - Removed Permission menu access on Permissions to role Admin  
[2025-10-18 21:30:21,774] {manager.py:543} INFO - Removed Permission View: menu_access on Permissions  
[2025-10-18 21:30:21,834] {manager.py:508} INFO - Created Permission view: menu_access on Permissions  
[2025-10-18 21:30:21,844] {manager.py:568} INFO - Added Permission menu access on Permissions to role Admin  
hadop@615cf53bef6c:~/airflow/dags$  
hadop@615cf53bef6c:~/airflow/dags$
```

El DAG se llama ***ingest-transform-2***:

The screenshot shows the Apache Airflow web interface in a browser window at localhost:8010/home. The interface includes a top navigation bar with the Airflow logo and links for DAGs, Security, Browse, Admin, and Docs. The right side of the header shows the current time as 00:51 UTC and a user profile icon. The main content area displays a table of DAGs:

DAG Name	Category	Frequency	Last Run	Next Run	Status	Actions
example_weekday_branch_operator	example	airflow	@daily	2025-10-18, 00:00:00	Running	▶ 🗑️ ⋮
example_xcom	example	airflow	@once	2021-01-01, 00:00:00	Failed	▶ 🗑️ ⋮
example_xcom_args	example	airflow	None		Failed	▶ 🗑️ ⋮
example_xcom_args_with_operators	example	airflow	None		Failed	▶ 🗑️ ⋮
ingest-transform	ingest / transform	airflow	00:00:00	2022-05-08, 00:00:00	Running	▶ 🗑️ ⋮
ingest-transform-2	ingest / transform	airflow	00:00:00	2025-10-17, 00:00:00	Running	▶ 🗑️ ⋮

