EXAMEN FINAL – BOOTCAMP DATA ENGINEERING

ALUMNO: MATIAS EZEQUIEL PADILLA PRESAS

EJERCICIO 1

Aviación Civil

La Administración Nacional de Aviación Civil necesita una serie de informes para elevar al ministerio de transporte acerca de los aterrizajes y despegues en todo el territorio Argentino, como puede ser: cuales aviones son los que más volaron, cuántos pasajeros volaron, ciudades de partidas y aterrizajes entre fechas determinadas, etc.
Usted como data engineer deberá realizar un pipeline con esta información, automatizarlo y realizar los análisis de datos solicitados que permita responder las preguntas de negocio, y hacer sus recomendaciones con respecto al estado actual.

Listado de vuelos realizados:
https://datos.gob.ar/lv/dataset/transporte-aterrizajes-despegues-procesados-por-administracion-nacional-aviacion-civil-anac

Listado de detalles de aeropuertos de Argentina:
https://datos.transporte.gob.ar/dataset/lista-aeropuertos

TAREAS

1. Hacer ingest de los siguientes files relacionados con transporte aéreo de Argentina :

2021:

https://dataengineerpublic.blob.core.windows.net/data-engineer/2021-informe-ministerio.csv

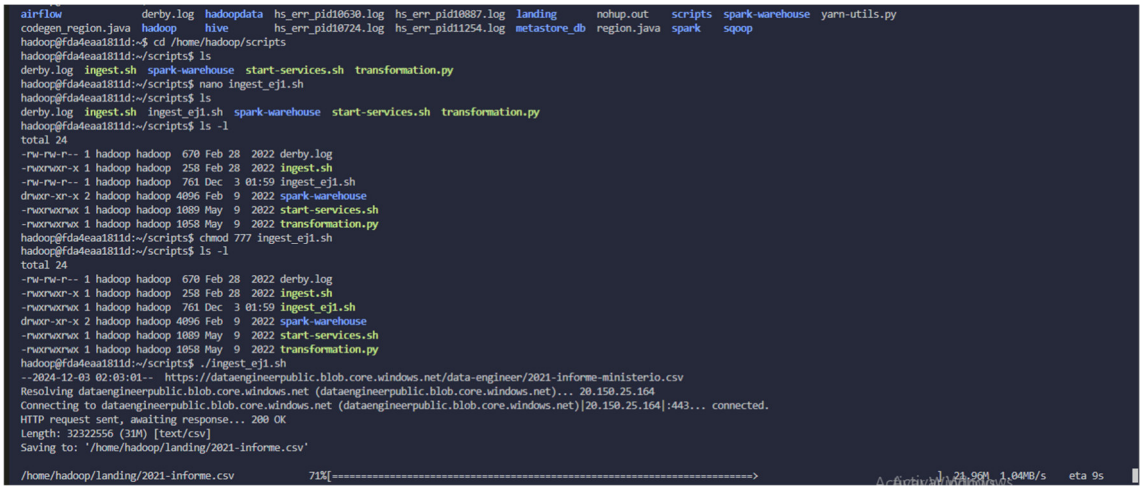
2022:

https://dataengineerpublic.blob.core.windows.net/data-engineer/202206-informe-ministerio.csv

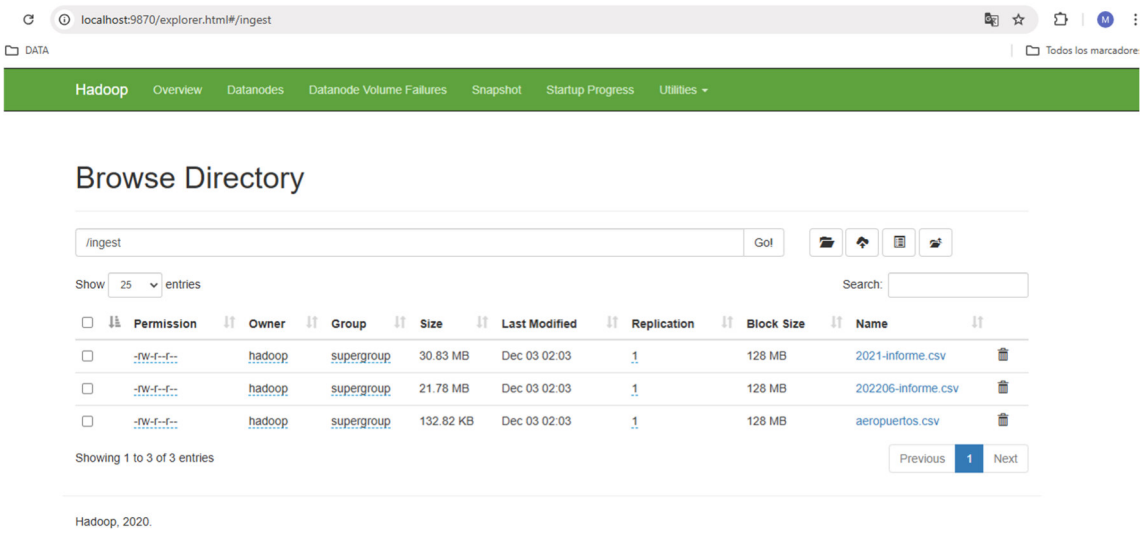
Aeropuertos_detalles:

https://dataengineerpublic.blob.core.windows.net/data-engineer/aeropuertos_detalle.csv

Creo el script que va a ingestar los 3 archivos csv en HDFS. Descargo los archivos con **wget** y los envio a un directorio **landing,** para luego con la orden hdfs dfs -put ingestarlos en el directorio **ingest** de HDFS. Antes de correr el script para verificar su funcionamiento debo otorgarle los permisos necesarios.



Chequeo que los archivos se encuentren en HDFS



2. Crear 2 tablas en el datawarehouse, una para los vuelos realizados en 2021 y 2022 (2021-informe-ministerio.csv y 202206-informe-ministerio) y otra tabla para el detalle de los aeropuertos (aeropuertos_detalle.csv)

Schema Tabla 1:

campos	tipo
fecha	date
horaUTC	string
clase_de_vuelo	string
clasificacion_de_vuelo	string
tipo_de_movimiento	string
aeropuerto	string
origen_destino	string
aerolinea_nombre	string
aeronave	string
pasajeros	integer

Creo una **DATABASE** en Hive para alojar las tablas que voy a crear.

```
SLF4J: Found binding in [jar:file:/home/hadoop/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/hadoop/hive/lib/hive-common-2.3.9.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using
Hive 1.X releases.
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.hive.common.StringInternUtils (file:/home/hadoop/hive/lib/hive-common-2.3.9.jar) to field java.net.
URI.string
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.hive.common.StringInternUtils
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
hive> show databases;
OK
default
ejercicio1db
tripdata
Time taken: 13.855 seconds, Fetched: 3 row(s)
hive> create database db_ej1
    > ;
OK
Time taken: 0.568 seconds
```

Creo las tablas con las que vamos a trabajar

```
PROBLEMS 6  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

    > create external table db_ej1.h_info_tab (
    > fecha date,
    > horaUTC string,
    > clase_de_vuelo string,
    > clasificacion_de_vuelo string,
    > tipo_de_movimiento string,
    > aeropuerto string,
    > origen_destino string,
    > aerolinea_nombre string,
    > aeronave string,
    > pasajeros int
    > )
    > row format delimited
    > fields terminated by ";"
    > ;
OK
Time taken: 0.537 seconds
```

```
hive> CREATE EXTERNAL TABLE db_ej1.h_aerop_tab (
    > aeropuerto string,
    > oac string,
    > iata string,
    > tipo string,
    > denominacion string,
    > coordenadas string,
    > latitud string,
    > longitud string,
    > elev float,
    > uom_elev string,
    > ref string,
    > distancia_ref float,
    > direccion_ref string,
    > condicion string,
    > control string,
    > region string,
    > uso string,
    > trafico string,
    > sna string,
    > concesionado string,
    > provincia string
    > )
    > row format delimited
    > FIELDS TERMINATED BY ','
    > ;
OK
Time taken: 0.085 seconds
```

3. Realizar un proceso automático orquestado por airflow que ingeste los archivos previamente menc onados entre las fechas 01/01/2021 y 30/06/2022 en las dos columnas creadas.

   Los archivos 202206-informe-ministerio.csv y 202206-informe-ministerio.csv → en la tabla aeropuerto_tabla

   El archivo aeropuertos_detalle.csv → en la tabla aeropuerto_detalles_tabla

4. Realizar las siguiente transformaciones en los pipelines de datos:
   - Eliminar la columna inhab ya que no se utilizará para el análisis
   - Eliminar la columna fir ya que no se utilizará para el análisis
   - Eliminar la columna "calidad del dato" ya que no se utilizará para el análisis
   - Filtrar los vuelos internacionales ya que solamente se analizarán los vuelos domésticos
   - En el campo pasajeros si se encuentran campos en Null convertirlos en 0 (cero)
   - En el campo distancia_ref si se encuentran campos en Null convertirlos en 0 (cero)

Utilizo PySpark para comprobar el código que se correrá en el proceso orquestado con Airflow.

Creo los DF y verifico el Schema , hago un “show” para verificar el una muestra de las tablas

```
>>> df_inf21 = (
...     spark.read.option("header", "true")
...     .option("delimiter", ";")
...     .csv("hdfs://172.17.0.2:9000/ingest/2021-informe.csv")
... )
```

```
>>> df_inf22 = (
...     spark.read.option("header", "true")
...     .option("delimiter", ";")
...     .csv("hdfs://172.17.0.2:9000/ingest/202206-informe.csv")
... )
```

```
>>> df_aerop = (
...     spark.read.option("header", "true")
...     .option("delimiter", ";")
...     .csv("hdfs://172.17.0.2:9000/ingest/aeropuertos.csv")
... )
```

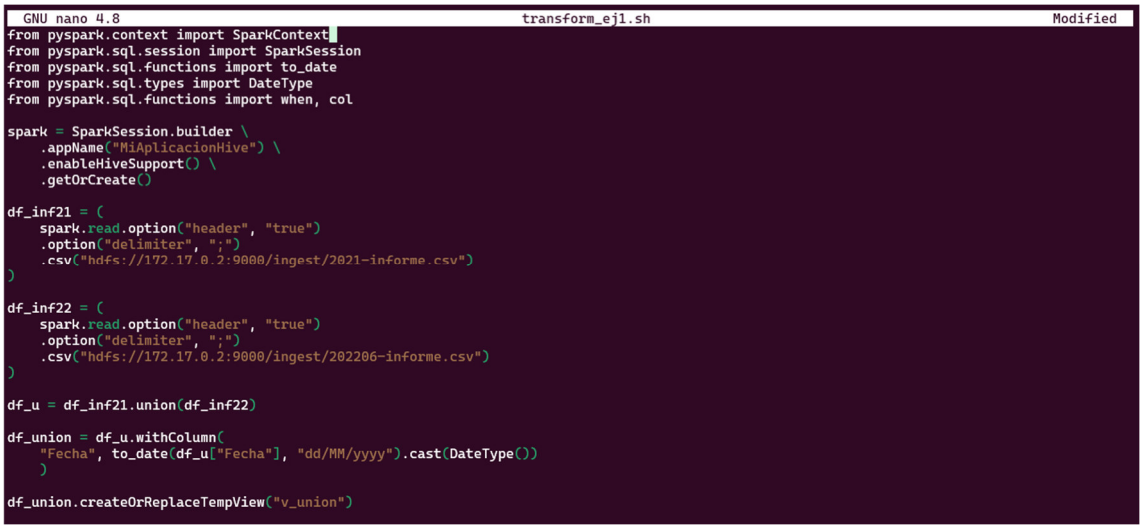
Genero las vistas y hago las transformaciones y cargo en las tablas previamente creadas en Hive

```
>>> df_inf21.count()
328136
>>> df_inf22.count()
222927
>>> df_aerop.count()
693
>>> df_u = df_inf21.union(df_inf22)
>>> df_u.count()
551063
>>> df_union = df_u.withColumn(
...     "Fecha", to_date(df_u["Fecha"], "dd/MM/yyyy").cast(DateType())
... )
>>> df_union.createOrReplaceTempView("v_union")
```

```
>>> df_info_union = spark.sql("""
...     SELECT
...         CAST(Fecha AS DATE) AS fecha,
...         CAST(`Hora UTC` AS STRING) AS horaUTC,
...         CAST(`Clase de Vuelo (todos los vuelos)` AS STRING) AS clase_de_vuelo,
...         CAST(`Clasificación Vuelo` AS STRING) AS clasificacion_de_vuelo,
...         CAST(`Tipo de Movimiento` AS STRING) AS tipo_de_movimiento,
...         CAST(Aeropuerto AS STRING) AS aeropuerto,
...         CAST(`Origen / Destino` AS STRING) AS origen_destino,
...         CAST(`Aerolinea Nombre` AS STRING) AS aerolinea_nombre,
...         CAST(Aeronave AS STRING) AS aeronave,
...         CAST(Pasajeros AS INTEGER) AS pasajeros
...     FROM v_union
...     WHERE `Clasificación Vuelo` <> 'Internacional'
... """)
>>> df_info_notnull = df_info_union.withColumn(
...     "pasajeros", when(col("pasajeros").isNull(), 0).otherwise(col("pasajeros"))
... )
>>> df_info_notnull.createOrReplaceTempView("info_tabla_view")
>>> spark.sql("INSERT INTO db_ej1.h_info_tab SELECT * FROM info_tabla_view")
DataFrame[]
>>>
```

```
>>> df_aerop.createOrReplaceTempView("v_aerop")
>>> df_aerop_tabla = spark.sql("""
...     SELECT
...         CAST(local AS STRING) AS aeropuerto,
...         CAST(oaci AS STRING) AS oaci,
...         CAST(iata AS STRING) AS iata,
...         CAST(tipo AS STRING) AS tipo,
...         CAST(denominacion AS STRING) AS denominacion,
...         CAST(coordenadas AS STRING) AS coordenadas,
...         CAST(longitud AS STRING) AS longitud,
...         CAST(latitud AS STRING) AS latitud,
...         CAST(elev AS FLOAT) AS elev,
...         CAST(uom_elev AS STRING) AS uom_elev,
...         CAST(ref AS INTEGER) AS ref,
...         CAST(distancia_ref AS FLOAT) AS distancia_ref,
...         CAST(direccion_ref AS STRING) AS direccion_ref,
...         CAST(condicion AS STRING) AS condicion,
...         CAST(control AS STRING) AS control,
...         CAST(region AS STRING) AS region,
...         CAST(uso AS STRING) AS uso,
...         CAST(trafico AS STRING) AS trafico,
...         CAST(sna AS STRING) AS sna,
...         CAST(concesionado AS STRING) AS concesionado,
...         CAST(provincia AS STRING) AS provincia
...     FROM v_aerop
... """)
>>> df_aerop_filtrado = df_aerop_tabla.withColumn(
...     "distancia_ref", when(col("distancia_ref").isNull(), 0).otherwise(col("distancia_ref"))
... )
>>> df_aerop_filtrado.createOrReplaceTempView("aero_view")
>>> spark.sql("INSERT INTO db_ej1.h_aerop_tabla SELECT * FROM aero_view")
2024-12-30 15:58:11,275 WARN conf.HiveConf: HiveConf of name hive.metastore.local does not exist
2024-12-30 15:58:11,731 WARN session.SessionState: METASTORE_FILTER_HOOK will be ignored, since hive.security.authorization.manager is set t
o instance of HiveAuthorizerFactory.
DataFrame[]
```

Con este código creo el script de transformación que luego correré en un DAG de



```
 GNU nano 4.8                          transform_ej1.sh                          Modified
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql.functions import to_date
from pyspark.sql.types import DateType
from pyspark.sql.functions import when, col

spark = SparkSession.builder \
    .appName("MiAplicacionHive") \
    .enableHiveSupport() \
    .getOrCreate()

df_inf21 = (
    spark.read.option("header", "true")
    .option("delimiter", ";")
    .csv("hdfs://172.17.0.2:9000/ingest/2021-informe.csv")
)

df_inf22 = (
    spark.read.option("header", "true")
    .option("delimiter", ";")
    .csv("hdfs://172.17.0.2:9000/ingest/202206-informe.csv")
)

df_u = df_inf21.union(df_inf22)

df_union = df_u.withColumn(
    "Fecha", to_date(df_u["Fecha"], "dd/MM/yyyy").cast(DateType())
    )

df_union.createOrReplaceTempView("v_union")
```

```
 GNU nano 4.8                          transform_ej1.sh                          Modified

df_info_union = spark.sql("""
    SELECT
        CAST(Fecha AS DATE) AS fecha,
        CAST(`Hora UTC` AS STRING) AS horaUTC,
        CAST(`Clase de Vuelo (todos los vuelos)` AS STRING) AS clase_de_vuelo,
        CAST(`Clasificaci      n Vuelo` AS STRING) AS clasificacion_de_vuelo,
        CAST(`Tipo de Movimiento` AS STRING) AS tipo_de_movimiento,
        CAST(Aeropuerto AS STRING) AS aeropuerto,
        CAST(`Origen / Destino` AS STRING) AS origen_destino,
        CAST(`Aerolinea Nombre` AS STRING) AS aerolinea_nombre,
        CAST(Aeronave AS STRING) AS aeronave,
        CAST(Pasajeros AS INTEGER) AS pasajeros
    FROM v_union
    WHERE `Clasificaci      n Vuelo` <> 'Internacional'
""")

df_info_notnull = df_info_union.withColumn(
    "pasajeros", when(col("pasajeros").isNull(), 0).otherwise(col("pasajeros"))
)

df_info_notnull.createOrReplaceTempView("info_tabla_view")

spark.sql("INSERT INTO db_ej1.h_info_tab SELECT * FROM info_tabla_view")


df_aerop = (
    spark.read.option("header", "true")
    .option("delimiter", ";")
    .csv("hdfs://172.17.0.2:9000/ingest/aeropuertos.csv")
)

df_aerop.createOrReplaceTempView("v_aerop")
```

```
 GNU nano 4.8                          transform_ej1.sh                          Modified
df_aerop.createOrReplaceTempView("v_aerop")

df_aerop_tabla = spark.sql("""
    SELECT
        CAST(local AS STRING) AS aeropuerto,
        CAST(oaci AS STRING) AS oaci,
        CAST(iata AS STRING) AS iata,
        CAST(tipo AS STRING) AS tipo,
        CAST(denominacion AS STRING) AS denominacion,
        CAST(coordenadas AS STRING) AS coordenadas,
        CAST(longitud AS STRING) AS longitud,
        CAST(latitud AS STRING) AS latitud,
        CAST(elev AS FLOAT) AS elev,
        CAST(uom_elev AS STRING) AS uom_elev,
        CAST(ref AS INTEGER) AS ref,
        CAST(distancia_ref AS FLOAT) AS distancia_ref,
        CAST(direccion_ref AS STRING) AS direccion_ref,
        CAST(condicion AS STRING) AS condicion,
        CAST(control AS STRING) AS control,
        CAST(region AS STRING) AS region,
        CAST(uso AS STRING) AS uso,
        CAST(trafico AS STRING) AS trafico,
        CAST(sna AS STRING) AS sna,
        CAST(concesionado AS STRING) AS concesionado,
        CAST(provincia AS STRING) AS provincia
    FROM v_aerop
""")

df_aerop_filtrado = df_aerop_tabla.withColumn(
    "distancia_ref", when(col("distancia_ref").isNull(), 0).otherwise(col("distancia_ref"))
)

df_aerop_filtrado.createOrReplaceTempView("aero_view")

spark.sql("INSERT INTO db_ej1.h_aerop_tabla SELECT * FROM aero_view")
```

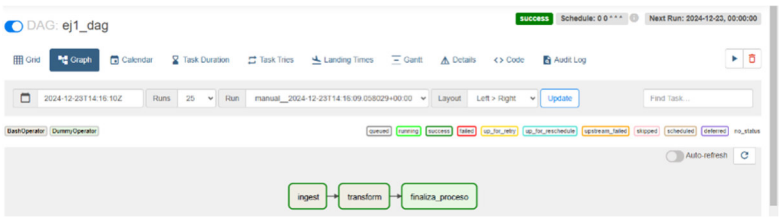
Airflow

Le doy permisos de ejecución al script

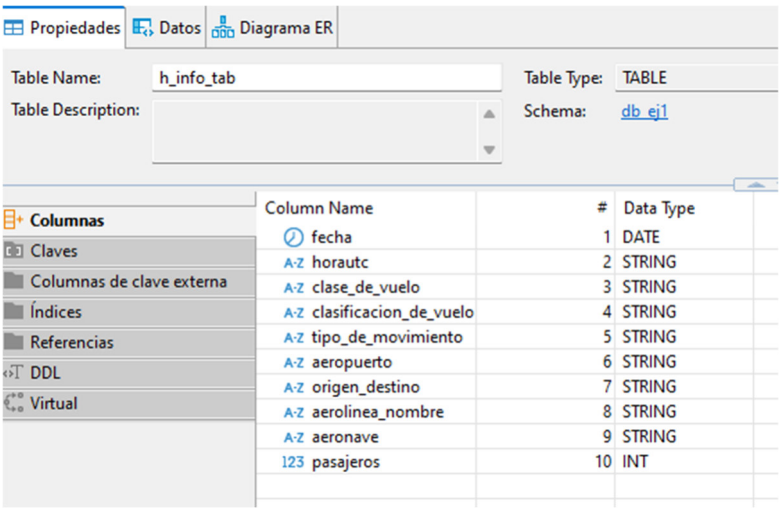
```
hadoop@fda4eaa1811d:/$ cd home/hadoop/scripts
hadoop@fda4eaa1811d:~/scripts$ ls
derby.log  ingest.sh  ingest_ej1.sh  spark-warehouse  start-services.sh  transform_ej1.py  transformation.py
hadoop@fda4eaa1811d:~/scripts$ nano ingest_ej1.sh
hadoop@fda4eaa1811d:~/scripts$ nano transform_ej1.py
hadoop@fda4eaa1811d:~/scripts$ nano transform_ej1.py
hadoop@fda4eaa1811d:~/scripts$ chmod 777 transformation.py
hadoop@fda4eaa1811d:~/scripts$ ls -l
total 28
-rw-rw-r-- 1 hadoop hadoop  670 Feb 28  2022 derby.log
-rwxrwxr-x 1 hadoop hadoop  258 Feb 28  2022 ingest.sh
-rwxrwxrwx 1 hadoop hadoop  761 Dec  3 01:59 ingest_ej1.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 3195 Dec 18 06:01 transform_ej1.py
-rwxrwxrwx 1 hadoop hadoop 1058 May  9  2022 transformation.py
hadoop@fda4eaa1811d:~/scripts$
```

Creo el dag y ya puedo correrlo en airflow.

```
 GNU nano 4.8                                    ej1_dag.py                                    Modifi
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

# Definicion de  argumentos por defecto
args = {
    'owner': 'airflow',
}

# Definicion del DAG
with DAG(
    dag_id='ej1_dag',
    default_args=args,
    schedule_interval='0 0 * * *',
    start_date=days_ago(2),
    dagrun_timeout=timedelta(minutes=60),
) as dag:

    # Tarea de finalizacon del proceso
    finaliza_proceso = DummyOperator(
        task_id='finaliza_proceso',
    )

    # Tarea de ingesta
    ingest = BashOperator(
        task_id='ingest',
        bash_command='/usr/bin/sh /home/hadoop/scripts/ingest_ej1.sh ',
    )

    # Tarea de transformacion
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/transform_ej1.py',
)


    # Definicion flujo de tareas
ingest >> transform >> finaliza_proceso
                                                                          Activar Windows
```



5. Mostrar mediante una impresión de pantalla, que los tipos de campos de las tablas sean los solicitados en el datawarehouse (ej: fecha date, aeronave string, pasajeros integer, etc.)

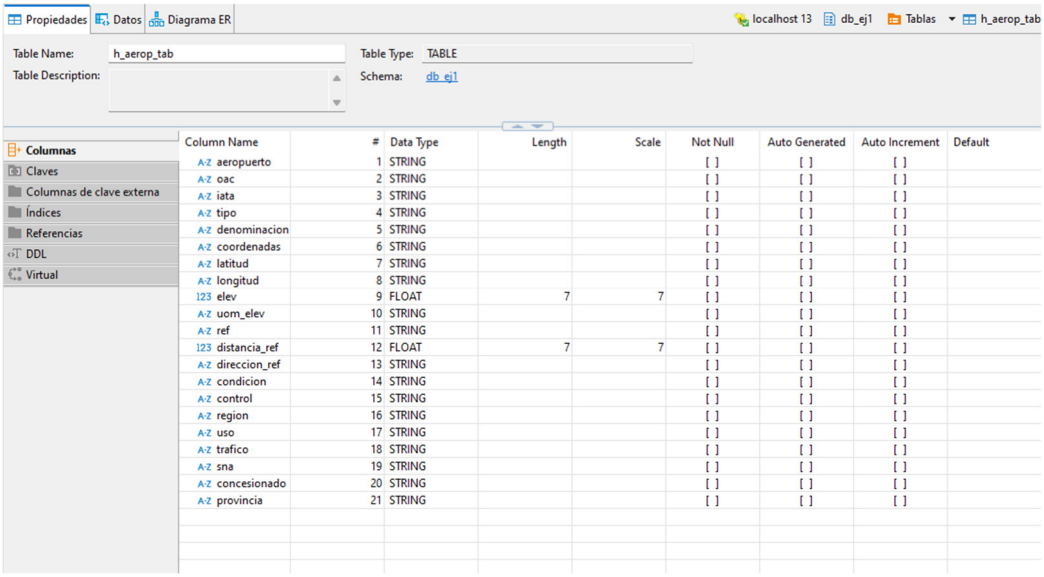


Propiedades | Datos | Diagrama ER

Table Name: h_info_tab
Table Description:
Table Type: TABLE
Schema: db_ej1

Columnas | Claves | Columnas de clave externa | Índices | Referencias | DDL | Virtual

Column Name	#	Data Type
fecha	1	DATE
horautc	2	STRING
clase_de_vuelo	3	STRING
clasificacion_de_vuelo	4	STRING
tipo_de_movimiento	5	STRING
aeropuerto	6	STRING
origen_destino	7	STRING
aerolinea_nombre	8	STRING
aeronave	9	STRING
pasajeros	10	INT



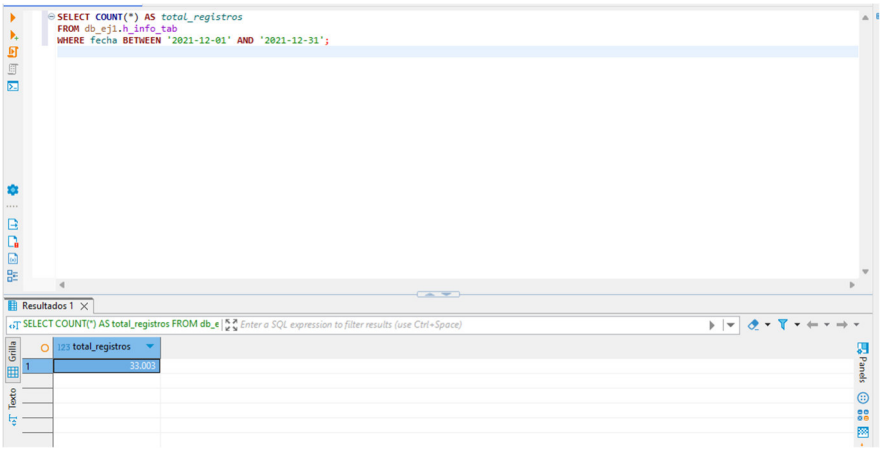
Propiedades | Datos | Diagrama ER — localhost 13 | db_ej1 | Tablas | h_aerop_tab

Table Name: h_aerop_tab
Table Description:
Table Type: TABLE
Schema: db_ej1

Columnas | Claves | Columnas de clave externa | Índices | Referencias | DDL | Virtual

Column Name	#	Data Type	Length	Scale	Not Null	Auto Generated	Auto Increment	Default
aeropuerto	1	STRING			[ ]	[ ]	[ ]	
oac	2	STRING			[ ]	[ ]	[ ]	
iata	3	STRING			[ ]	[ ]	[ ]	
tipo	4	STRING			[ ]	[ ]	[ ]	
denominacion	5	STRING			[ ]	[ ]	[ ]	
coordenadas	6	STRING			[ ]	[ ]	[ ]	
latitud	7	STRING			[ ]	[ ]	[ ]	
longitud	8	STRING			[ ]	[ ]	[ ]	
elev	9	FLOAT	7	7	[ ]	[ ]	[ ]	
uom_elev	10	STRING			[ ]	[ ]	[ ]	
ref	11	STRING			[ ]	[ ]	[ ]	
distancia_ref	12	FLOAT	7	7	[ ]	[ ]	[ ]	
direccion_ref	13	STRING			[ ]	[ ]	[ ]	
condicion	14	STRING			[ ]	[ ]	[ ]	
control	15	STRING			[ ]	[ ]	[ ]	
region	16	STRING			[ ]	[ ]	[ ]	
uso	17	STRING			[ ]	[ ]	[ ]	
trafico	18	STRING			[ ]	[ ]	[ ]	
sna	19	STRING			[ ]	[ ]	[ ]	
concesionado	20	STRING			[ ]	[ ]	[ ]	
provincia	21	STRING			[ ]	[ ]	[ ]	

6. Determinar la cantidad de vuelos entre las fechas 01/12/2021 y 31/01/2022. Mostrar consulta y Resultado de la query



```
SELECT COUNT(*) AS total_registros
  FROM db_ej1.h_info_tab
  WHERE fecha BETWEEN '2021-12-01' AND '2021-12-31';
```

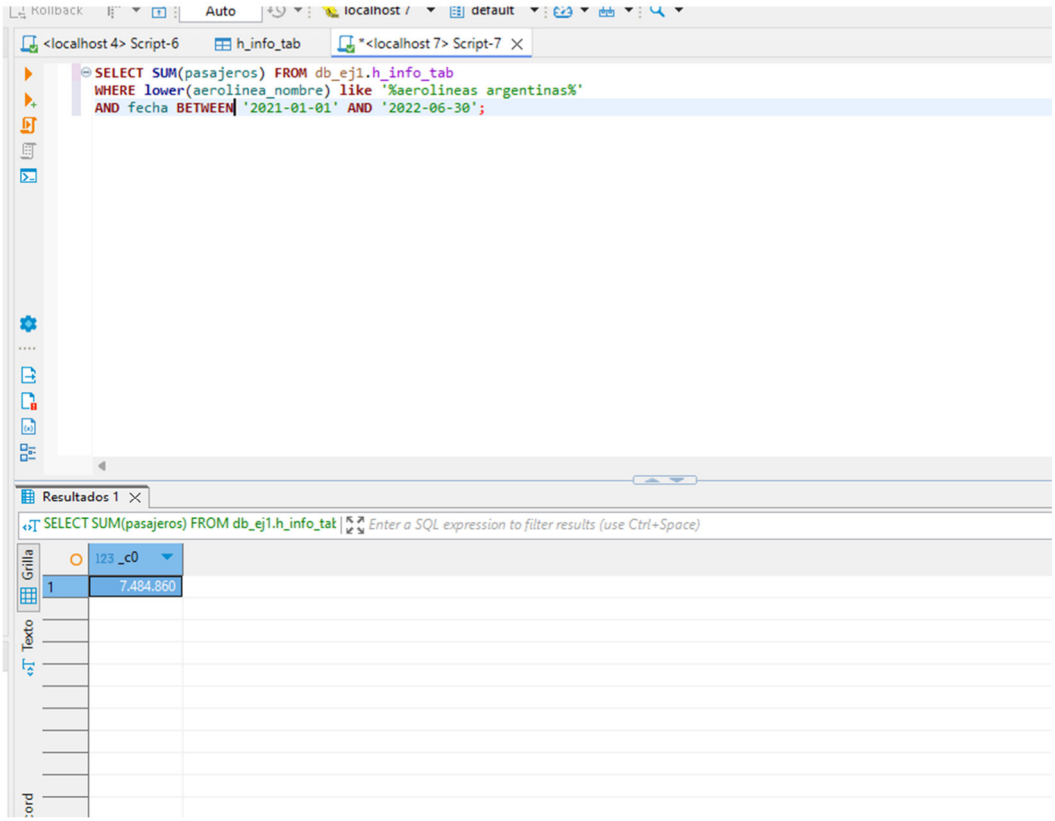
Resultados 1

SELECT COUNT(*) AS total_registros FROM db_e | Enter a SQL expression to filter results (use Ctrl+Space)

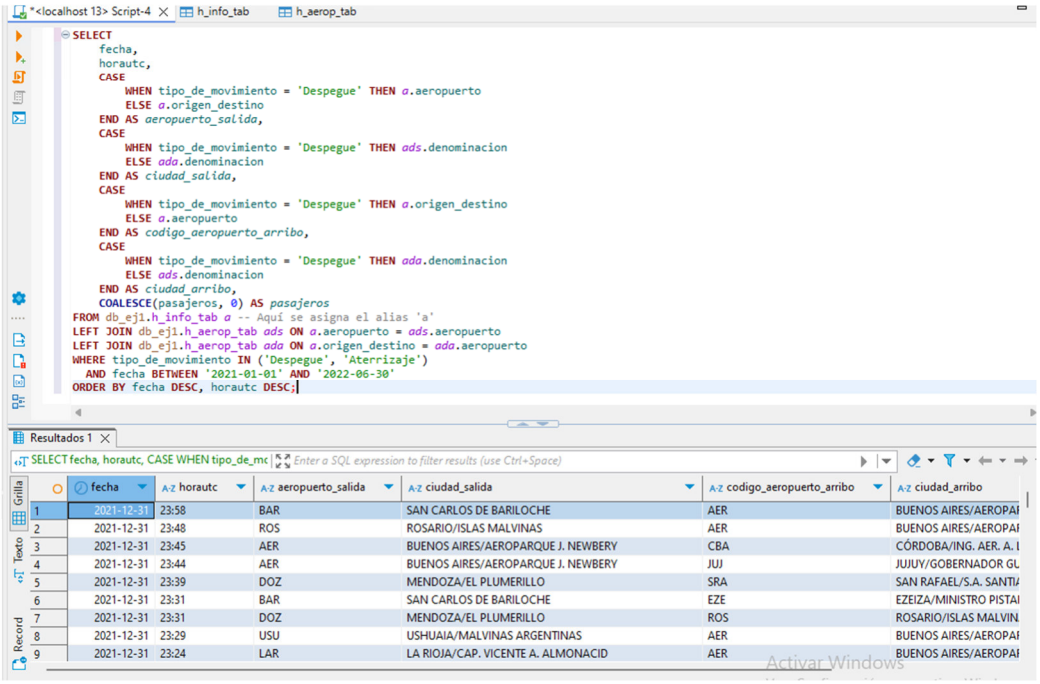
	123 total_registros
1	33.003

Grilla | Texto | Panels

7. Cantidad de pasajeros que viajaron en Aerolíneas Argentinas entre el 01/01/2021 y 30/06/2022. Mostrar consulta y Resultado de la query



8. Mostrar fecha, hora, código aeropuerto salida, ciudad de salida, código de aeropuerto de arribo, ciudad de arribo, y cantidad de pasajeros de cada vuelo, entre el 01/01/2022 y el 30/06/2022 ordenados por fecha de manera descendiente. Mostrar consulta y Resultado de la query

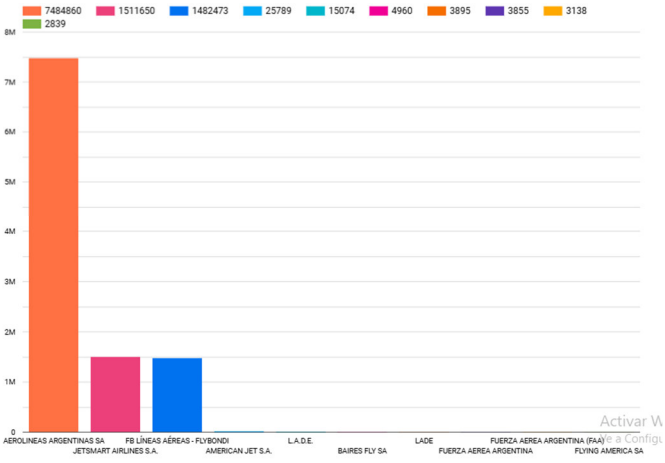


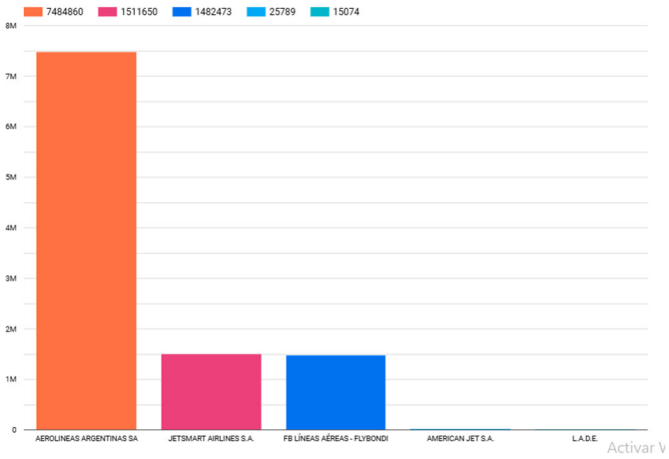
9. Cuales son las 10 aerolíneas que más pasajeros llevaron entre el 01/01/2021 y el 30/06/2022 exceptuando aquellas aerolíneas que no tengan nombre. Mostrar consulta y Visualización

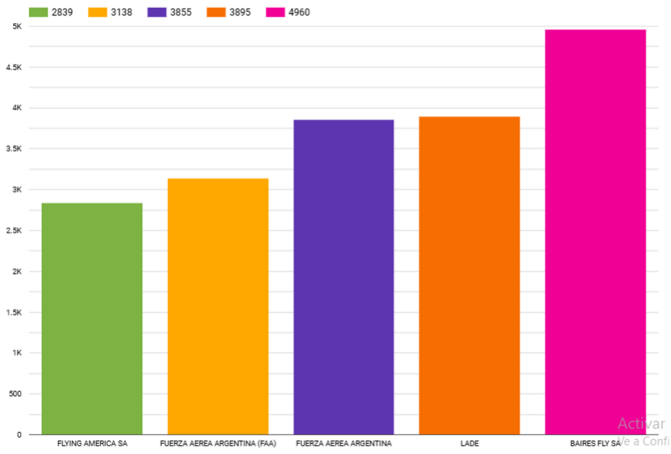
```
select aerolinea_nombre,
sum(pasajeros) total_pasajeros
from db_ej1.h_info_tab
where aerolinea_nombre is not NULL and aerolinea_nombre <> '0'
and fecha between '2021-01-01' and '2022-06-30'
group by aerolinea_nombre
order by total_pasajeros desc limit 10;
```

Resultados 1

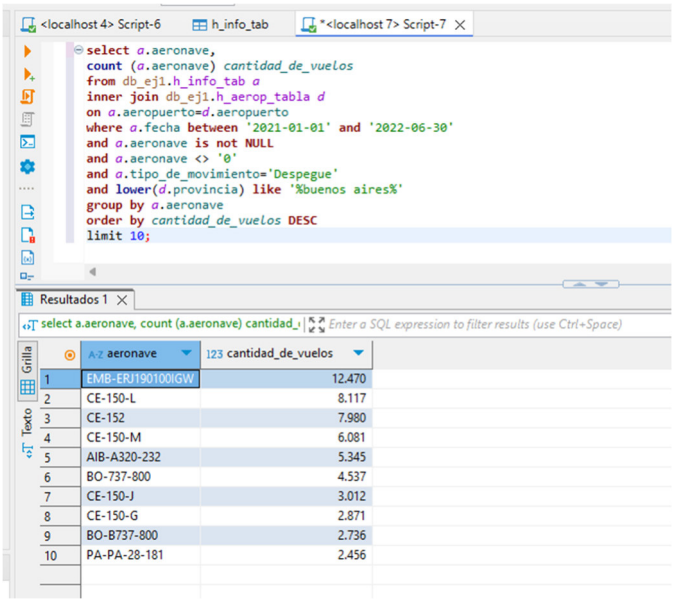
	aerolinea_nombre	total_pasajeros
1	AEROLINEAS ARGENTINAS SA	7.484.860
2	JETSMART AIRLINES S.A.	1.511.650
3	FB LÍNEAS AÉREAS - FLYBONDI	1.482.473
4	AMERICAN JET S.A.	25.789
5	L.A.D.E.	15.074
6	BAIRES FLY SA	4.960
7	LADE	3.895
8	FUERZA AEREA ARGENTINA	3.855
9	FUERZA AEREA ARGENTINA (FAA)	3.138
10	FLYING AMERICA SA	2.839







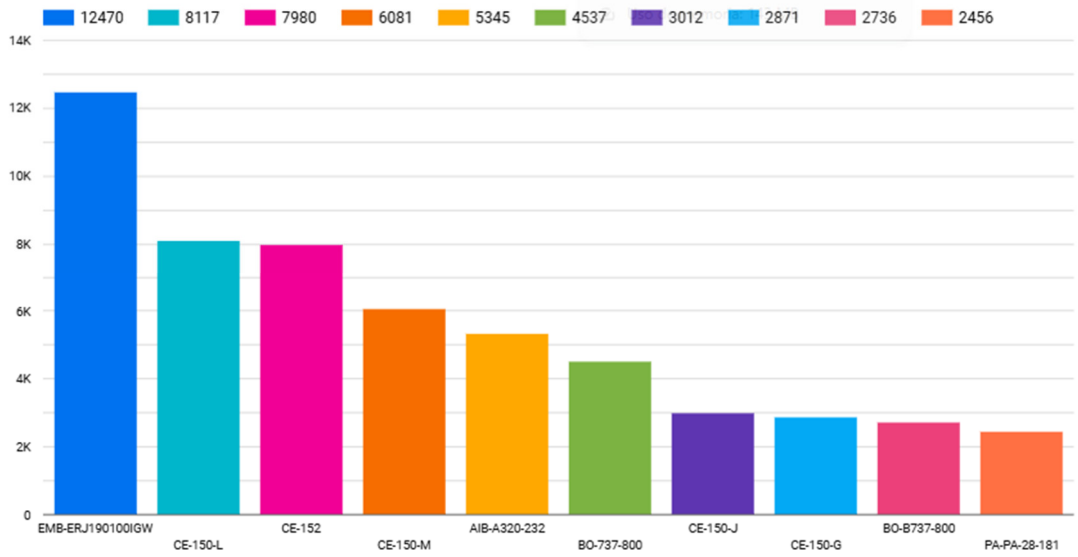
10. Cuales son las 10 aeronaves más utilizadas entre el 01/01/2021 y el 30/06/22 que despegaron desde la Ciudad autónoma de Buenos Aires o de Buenos Aires, exceptuando aquellas aeronaves que no cuentan con nombre. Mostrar consulta y Visualización



```
select a.aeronave,
count (a.aeronave) cantidad_de_vuelos
from db_ej1.h_info_tab a
inner join db_ej1.h_aerop_tabla d
on a.aeropuerto=d.aeropuerto
where a.fecha between '2021-01-01' and '2022-06-30'
and a.aeronave is not NULL
and a.aeronave <> '0'
and a.tipo_de_movimiento='Despegue'
and lower(d.provincia) like '%buenos aires%'
group by a.aeronave
order by cantidad_de_vuelos DESC
limit 10;
```

Resultados 1

	aeronave	cantidad_de_vuelos
1	EMB-ERJ190100IGW	12.470
2	CE-150-L	8.117
3	CE-152	7.980
4	CE-150-M	6.081
5	AIB-A320-232	5.345
6	BO-737-800	4.537
7	CE-150-J	3.012
8	CE-150-G	2.871
9	BO-B737-800	2.736
10	PA-PA-28-181	2.456



11. Qué datos externos agregaría en este dataset que mejoraría el análisis de los datos
12. Elabore sus conclusiones y recomendaciones sobre este proyecto.
13. Proponer una arquitectura alternativa para este proceso ya sea con herramientas on premise o cloud (Sí aplica)

11- Para las consultas propuestas agregaría algún dato sobre días festivos o elaboraría en base a la información disponible algún campo para identificar temporadas altas o bajas con respecto a la cantidad de viajeros según fechas.

12-Dada la diferencia de entre el tráfico que mueve Aerolíneas Argentinas, podríamos optar por aconsejar abrir nuevas rutas o aumentar la frecuencia de las existentes. En base a la cantidad de vuelos de las naves, podemos estimar la necesidad de mayor cantidad de controles, mantenimiento y hasta recambio de las mismas.

Se podría pensar en enriquecer la base cruzándola con otras que por ejemplo midan costos, ingresos y otro tipo de datos financieros para poder medir el impacto económico de estas métricas.

Debido a la creciente continua en la cantidad de vuelos que impacta en los datos que alimentan la base, deberíamos optar por particionar la tabla por fecha para facilitar y hacer mas eficiente las consultas sobre la misma.

Hacer hincapié en la calidad de los datos, la forma de ingresarlos y la completitud de los mismos.

13- Propondría una arquitectura en el entorno cloud de Google (GCP). Principalmente por la disponibilidad online, la capacidad de procesamiento y la robustez.

Utilizaría para ingesta Google Cloud Storage, el procesamiento ETL con Dataflow y BigQuery para el almacenamiento. En acaso de automatizar y orquestar el proceso, sería a través de Airflow con Cloud Composer y propondría utilizar Dataprep, para un procesamiento más fino de los datos con fines de enriquecer la base.

Ejercicio 2:

Alquiler de automóviles

Una de las empresas líderes en alquileres de automóviles solicita una serie de dashboards y reportes para poder basar sus decisiones en datos. Entre los indicadores mencionados se encuentran total de alquileres, segmentación por tipo de combustible, lugar, marca y modelo de automóvil, valoración de cada alquiler, etc.
Como Data Engineer debe crear y automatizar el pipeline para tener como resultado los datos listos para ser visualizados y responder las preguntas de negocio.

1. Crear en hive una database car_rental_db y dentro una tabla llamada car_rental_analytics, con estos campos:

campos	tipo
fuelType	string

```
hive> show databases;
OK
db_ej1
default
ejercicio1db
tripdata
Time taken: 0.163 seconds, Fetched: 4 row(s)
hive> create database car_rental_db;
OK
Time taken: 0.577 seconds
hive> use car_rental_db;
OK
Time taken: 0.031 seconds
```

rating	integer
renterTripsTaken	integer
reviewCount	integer
city	string
state_name	string
owner_id	integer
rate_daily	integer
make	string
model	string
year	integer

```
hive> use car_rental_db;
OK
Time taken: 0.031 seconds
hive> CREATE EXTERNAL TABLE car_rental_db.car_rental_analytics (
    >     fuelType STRING,
    >     rating INT,
    >     renterTripsTaken INT,
    >     reviewCount INT,
    >     city STRING,
    >     state_name STRING,
    >     owner_id INT,
    >     rate_daily INT,
    >     rate_daily INT,
    >     make STRING,
    >     model STRING,
    >     year INT,
    >     geo_point STRING,
    >     geo_shape STRING,
    >     official_code_state STRING,
    >     official_name_state STRING,
    >     iso_3166_3_area_code STRING,
    >     type STRING,
    >     usps_state_abbreviation STRING,
    >     state_fips_code STRING,
    >     state_gnis_code STRING
    > )
    > row format delimited
    > fields terminated by ";"
    > ;
OK
Time taken: 0.568 seconds
hive> show tables;
OK
car_rental_analytics
Time taken: 0.036 seconds, Fetched: 1 row(s)
hive>
```

2. Crear script para el ingest de estos dos files

https://dataengineerpublic.blob.core.windows.net/data-engineer/CarRentalData.csv

https://dataengineerpublic.blob.core.windows.net/data-engineer/georef-united-states-of-america-state.csv

Sugerencia: descargar el segundo archivo con un comando similar al abajo mencionado, ya que al tener caracteres como '&' falla si no se le asignan comillas. Adicionalmente, el parámetro -O permite asignarle un nombre más legible al archivo descargado

wget -P ruta_destino -O ruta_destino/nombre_archivo.csv ruta_al_archivo

Info del dataset: https://www.kaggle.com/datasets/kushleshkumar/cornell-car-rental-dataset

Creo el script de ingesta y le doy los permisos de ejecución

```
#Borro todos los archivos que puedan haber en la carpeta landing
rm -f /home/hadoop/landing/*.*

#Descargo los archivos en la carpeta landing
wget -O /home/hadoop/landing/car_rental.csv https://dataengineerpublic.blob.core.windows.net/data-engineer/CarRentalData.csv

wget -O /home/hadoop/landing/georef_usa.csv https://dataengineerpublic.blob.core.windows.net/data-engineer/georef-united-states-of-america-state.csv

#Borro los archivos presentes en HDFS /ingest
/home/hadoop/hadoop/bin/hdfs dfs -rm /ingest/*.*

#Muevo los archivos de landing a HDFS
/home/hadoop/hadoop/bin/hdfs dfs -put /home/hadoop/landing/*.* /ingest
```

```
root@DESKTOP-1000430:/mnt/c/Users/User# docker start edvai_hadoop
edvai_hadoop
root@DESKTOP-1000430:/mnt/c/Users/User# docker exec -it edvai_hadoop bash
root@fda4eaa1811d:/# su hadoop
hadoop@fda4eaa1811d:/$ cd home/hadoop/scripts
hadoop@fda4eaa1811d:~/scripts$ ls
derby.log  ingest.sh  ingest_ej1.sh  spark-warehouse  start-services.sh  transform_ej1.py  transformation.py
hadoop@fda4eaa1811d:~/scripts$ nano ingest_ej2.sh
hadoop@fda4eaa1811d:~/scripts$ ls -l
total 32
-rw-rw-r-- 1 hadoop hadoop  670 Feb 28  2022 derby.log
-rwxrwxr-x 1 hadoop hadoop  258 Feb 28  2022 ingest.sh
-rwxrwxrwx 1 hadoop hadoop  761 Dec  3 01:59 ingest_ej1.sh
-rw-rw-r-- 1 hadoop hadoop  623 Dec 20 04:11 ingest_ej2.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 3195 Dec 18 06:01 transform_ej1.py
-rwxrwxrwx 1 hadoop hadoop 1058 May  9  2022 transformation.py
hadoop@fda4eaa1811d:~/scripts$ chmod 777 ingest_ej2.sh
hadoop@fda4eaa1811d:~/scripts$ ls -l
total 32
-rw-rw-r-- 1 hadoop hadoop  670 Feb 28  2022 derby.log
-rwxrwxr-x 1 hadoop hadoop  258 Feb 28  2022 ingest.sh
-rwxrwxrwx 1 hadoop hadoop  761 Dec  3 01:59 ingest_ej1.sh
-rwxrwxrwx 1 hadoop hadoop  623 Dec 20 04:11 ingest_ej2.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 3195 Dec 18 06:01 transform_ej1.py
-rwxrwxrwx 1 hadoop hadoop 1058 May  9  2022 transformation.py
hadoop@fda4eaa1811d:~/scripts$
```

3. Crear un script para tomar el archivo desde HDFS y hacer las siguientes transformaciones:
   - En donde sea necesario, modificar los nombres de las columnas. Evitar espacios y puntos (reemplazar por _). Evitar nombres de columna largos
   - Redondear los float de 'rating' y castear a int.
   - Joinear ambos files
   - Eliminar los registros con rating nulo
   - Cambiar mayúsculas por minúsculas en 'fuelType'
   - Excluir el estado Texas

   Finalmente insertar en Hive el resultado

Pruebo todas las ordenes que voy a incluir en el script de transformación

```
>>> df_rental = (
...       spark.read.option("header", "true")
...       .option("delimiter", ",")
... .csv("hdfs://172.17.0.2:9000/ingest/car_rental.csv")
... )
>>> df_rental.printSchema()
root
 |-- fuelType: string (nullable = true)
 |-- rating: string (nullable = true)
 |-- renterTripsTaken: string (nullable = true)
 |-- reviewCount: string (nullable = true)
 |-- location.city: string (nullable = true)
 |-- location.country: string (nullable = true)
 |-- location.latitude: string (nullable = true)
 |-- location.longitude: string (nullable = true)
 |-- location.state: string (nullable = true)
 |-- owner.id: string (nullable = true)
 |-- rate.daily: string (nullable = true)
 |-- vehicle.make: string (nullable = true)
 |-- vehicle.model: string (nullable = true)
 |-- vehicle.type: string (nullable = true)
 |-- vehicle.year: string (nullable = true)
```

```
>>> df_rental.show(5)
+--------+------+----------------+-----------+-------------+----------------+----------------+----------------+-------------+-------+----------+------------+------------+----------
-+------------+
|fuelType|rating|renterTripsTaken|reviewCount|location.city|location.country|location.latitude|location.longitude|location.state|owner.id|rate.daily|vehicle.make|vehicle.model|vehicle.typ
e|vehicle.year|
+--------+------+----------------+-----------+-------------+----------------+----------------+----------------+-------------+-------+----------+------------+------------+----------
-+------------+
|ELECTRIC|     5|              13|         12|      Seattle|              US|       47.449107|      -122.308841|           WA|12847615|       135|       Tesla|     Model X|         su
v|        2019|
|ELECTRIC|     5|               2|          1|      Tijeras|              US|        35.11106|      -106.276551|           NM|15621242|       190|       Tesla|     Model X|         su
v|        2018|
|  HYBRID|  4.92|              28|         24|  Albuquerque|              US|       35.127163|      -106.566681|           NM|10199256|        35|      Toyota|       Prius|         ca
r|        2012|
|GASOLINE|     5|              21|         20|  Albuquerque|              US|       35.149726|      -106.711425|           NM| 9365496|        75|        Ford|     Mustang|         ca
r|        2018|
|GASOLINE|     5|               3|          1|  Albuquerque|              US|       35.208659|      -106.601008|           NM| 3553565|        47|    Chrysler|     Sebring|         ca
r|        2010|
+--------+------+----------------+-----------+-------------+----------------+----------------+----------------+-------------+-------+----------+------------+------------+----------
-+------------+
only showing top 5 rows

>>> df_rental.count()
5851
>>>
```



```
>>> df_rental.createOrReplaceTempView("v_rental")
>>> df_rental_mod = spark.sql("""
...     SELECT
...         CAST(fuelType AS STRING) AS fuelType,
...         CAST(rating AS INTEGER) AS rating,
...         CAST(renterTripsTaken AS INTEGER) AS renterTripsTaken,
...         CAST(reviewCount AS INTEGER) AS reviewCount,
...         CAST(`location.city` AS STRING) AS city,
...         CAST(`location.state` AS STRING) AS state_name,
...         CAST(`owner.id` AS INTEGER) AS owner_id,
...         CAST(`rate.daily` AS INTEGER) AS rate_daily,
...         CAST(`vehicle.make` AS STRING) AS make,
...         CAST(`vehicle.model` AS STRING) AS model,
...         CAST(`vehicle.year` AS INTEGER) AS year
...     FROM v_rental
... """)
>>>
```

```
>>> df_georef_mod = spark.sql("""
...     SELECT
...         CAST(`Geo Point` AS STRING) AS geo_point,
...         CAST(`Geo Shape` AS STRING) AS geo_shape,
...         CAST(`Year` AS STRING) AS year,
...         CAST(`Official Code State` AS STRING) AS official_code_state,
...         CAST(`Official Name State` AS STRING) AS official_name_state,
...         CAST(`Iso 3166-3 Area Code` AS STRING) AS iso_3166_3_area_code,
...         CAST(`Type` AS STRING) AS type,
...         CAST(`United States Postal Service state abbreviation` AS STRING) AS usps_state_abbreviation,
...         CAST(`State FIPS Code` AS STRING) AS state_fips_code,
...         CAST(`State GNIS Code` AS STRING) AS state_gnis_code
...     FROM v_georef
... """)
>>> df_merged = df_rental_mod.join(df_georef_mod, df_rental_mod.state_name == df_georef_mod.usps_state_abbreviation, "inner")
>>> df_merged.show(5)
+--------+------+----------------+-----------+----------+----------+-------+----------+------------+----------------+----+--------------------+--------------------+----+----------
----+--------------------+-----+--------------------+--------------+-------------+
|fuelType|rating|renterTripsTaken|reviewCount|      city|state_name|owner_id|rate_daily|        make|           model|year|           geo_point|           geo_shape|year|official_code_
state|official_name_state|iso_3166_3_area_code| type|usps_state_abbreviation|state_fips_code|state_gnis_code|
+--------+------+----------------+-----------+----------+----------+-------+----------+------------+----------------+----+--------------------+--------------------+----+----------
----+--------------------+-----+--------------------+--------------+-------------+
|    null|  null|               0|          0|San Antonio|        TX| 2737121|        25|       Mazda|         MAZDA2|2012|31.44720010145345...|"{""coordinates""...|2022|
   48|               Texas|                USA|state|                        TX|          null|       01779801|
|gasoline|     5|               1|          1|San Antonio|        TX|  812251|       271|  Land Rover|Range Rover Sport|2016|31.44720010145345...|"{""coordinates""...|2022|
   48|               Texas|                USA|state|                        TX|          null|       01779801|
|gasoline|     5|               1|          1|San Antonio|        TX| 1285519|        42|        Jeep|         Patriot|2015|31.44720010145345...|"{""coordinates""...|2022|
   48|               Texas|                USA|state|                        TX|          null|       01779801|
|gasoline|     4|               6|          5|San Antonio|        TX|12125275|        47|Mercedes-Benz|      GL-Class|2010|31.44720010145345...|"{""coordinates""...|2022|
   48|               Texas|                USA|state|                        TX|          null|       01779801|
|gasoline|  null|               0|          0|San Antonio|        TX|  812251|       195|     Cadillac|        Escalade|2015|31.44720010145345...|"{""coordinates""...|2022|
   48|               Texas|                USA|state|                        TX|          null|       01779801|
+--------+------+----------------+-----------+----------+----------+-------+----------+------------+----------------+----+--------------------+--------------------+----+----------
----+--------------------+-----+--------------------+--------------+-------------+
only showing top 5 rows
```

```
>>> df_rental_mod_clean = df_rental_mod.filter(df_rental_mod["rating"].isNotNull())
>>> from pyspark.sql.functions import lower
>>> df_rental_mod = df_rental_mod.withColumn("fuelType", lower(df_rental_mod["fuelType"]))
>>> df_rental_mod.show(5)
+--------+------+-----------------+-----------+-----------+----------+---------+----------+--------+-------+----+
|fuelType|rating|renterTripsTaken|reviewCount|       city|state_name| owner_id|rate_daily|    make|  model|year|
+--------+------+-----------------+-----------+-----------+----------+---------+----------+--------+-------+----+
|electric|     5|               13|         12|    Seattle|        WA| 12847615|       135|   Tesla|Model X|2019|
|electric|     5|                2|          1|    Tijeras|        NM| 15621242|       190|   Tesla|Model X|2018|
|  hybrid|     4|               28|         24|Albuquerque|        NM| 10199256|        35|  Toyota|  Prius|2012|
|gasoline|     5|               21|         20|Albuquerque|        NM|  9365496|        75|    Ford|Mustang|2018|
|gasoline|     5|                3|          1|Albuquerque|        NM|  3553565|        47|Chrysler|Sebring|2010|
+--------+------+-----------------+-----------+-----------+----------+---------+----------+--------+-------+----+
only showing top 5 rows

>>> df_georef = (
...         spark.read.option("header", "true")
...         .option("delimiter", ";")
...         .csv("hdfs://172.17.0.2:9000/ingest/georef_usa.csv")
... )
>>> df_georef.show(5)
+--------------------+--------------------+----+-------------------+-------------------+--------------------+-------------+-----------------------------------------------+---------------+
--------------+
|           Geo Point|           Geo Shape|Year|Official Code State|Official Name State|Iso 3166-3 Area Code|         Type|United States Postal Service state abbreviation|State FIPS Code|
State GNIS Code|
+--------------------+--------------------+----+-------------------+-------------------+--------------------+-------------+-----------------------------------------------+---------------+
--------------+
|31.44720010145345...|"{""coordinates""...|2022|                 48|              Texas|                 USA|        state|                                             TX|           null|
       01779801|
|38.64257169984573...|"{""coordinates""...|2022|                 54|      West Virginia|                 USA|        state|                                             WV|           null|
       01779805|
|18.21570685501268...|"{""coordinates""...|2022|                 72|        Puerto Rico|                 PRI|outlying area|                                             PR|           null|
       01779808|
|40.10998794109383...|"{""coordinates""...|2022|                 34|         New Jersey|                 USA|        state|                                             NJ|           null|
       01779795|
|20.99509455589766...|"{""coordinates""...|2022|                 15|             Hawaii|                 USA|        state|                                             HI|           null|
       01779782|
+--------------------+--------------------+----+-------------------+-------------------+--------------------+-------------+-----------------------------------------------+---------------+
--------------+
only showing top 5 rows

>>> df_georef.printSchema()
root
 |-- Geo Point: string (nullable = true)
 |-- Geo Shape: string (nullable = true)
 |-- Year: string (nullable = true)
 |-- Official Code State: string (nullable = true)
 |-- Official Name State: string (nullable = true)
 |-- Iso 3166-3 Area Code: string (nullable = true)
 |-- Type: string (nullable = true)
 |-- United States Postal Service state abbreviation: string (nullable = true)
 |-- State FIPS Code: string (nullable = true)
 |-- State GNIS Code: string (nullable = true)
```

```
>>> df_merged_filtered = df_merged.filter(df_merged.state_name != "TX")
>>> df_merged_filtered.createOrReplaceTempView("view_car_rental")
>>> spark.sql("""
... INSERT INTO TABLE car_rental_db.car_rental_analytics
... SELECT * FROM view_car_rental
... """)
2024-12-22 10:42:50,687 WARN conf.HiveConf: HiveConf of name hive.metastore.local does not exist
2024-12-22 10:42:51,084 WARN session.SessionState: METASTORE_FILTER_HOOK will be ignored, since hive.security.authorization.manager is set t
o instance of HiveAuthorizerFactory.
DataFrame[]
```

Creo el script y le doy los permisos necesarios

```
  GNU nano 4.8                          transform_ej2.py
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql.functions import to_date
from pyspark.sql.types import DateType
from pyspark.sql.functions import when, col
from pyspark.sql.functions import lower

spark = SparkSession.builder \
    .appName("MiAplicacionHive") \
    .enableHiveSupport() \
    .getOrCreate()
```

```
  GNU nano 4.8                          transform_ej2.py

df_rental_mod = spark.sql("""
    SELECT
        CAST(fuelType AS STRING) AS fuelType,
        CAST(rating AS INTEGER) AS rating,
        CAST(renterTripsTaken AS INTEGER) AS renterTripsTaken,
        CAST(reviewCount AS INTEGER) AS reviewCount,
        CAST(`location.city` AS STRING) AS city,
        CAST(`location.state` AS STRING) AS state_name,
        CAST(`owner.id` AS INTEGER) AS owner_id,
        CAST(`rate.daily` AS INTEGER) AS rate_daily,
        CAST(`vehicle.make` AS STRING) AS make,
        CAST(`vehicle.model` AS STRING) AS model,
        CAST(`vehicle.year` AS INTEGER) AS year
    FROM v_rental
""")

df_rental_mod_clean = df_rental_mod.filter(df_rental_mod["rating"].isNotNull())

df_rental_mod = df_rental_mod.withColumn("fuelType", lower(df_rental_mod["fuelType"]))

df_georef = (
        spark.read.option("header", "true")
        .option("delimiter", ";")
        .csv("hdfs://172.17.0.2:9000/ingest/georef_usa.csv")
)

df_georef.createOrReplaceTempView("v_georef")

df_georef_mod = spark.sql("""
    SELECT
        CAST(`Geo Point` AS STRING) AS geo_point,
```

```
 GNU nano 4.8                                                transform_ej2.py
df_georef = (
        spark.read.option("header", "true")
        .option("delimiter", ";")
        .csv("hdfs://172.17.0.2:9000/ingest/georef_usa.csv")
)

df_georef.createOrReplaceTempView("v_georef")

df_georef_mod = spark.sql("""
    SELECT
        CAST(`Geo Point` AS STRING) AS geo_point,
        CAST(`Geo Shape` AS STRING) AS geo_shape,
        CAST(`Official Code State` AS STRING) AS official_code_state,
        CAST(`Official Name State` AS STRING) AS official_name_state,
        CAST(`Iso 3166-3 Area Code` AS STRING) AS iso_3166_3_area_code,
        CAST(`Type` AS STRING) AS type,
        CAST(`United States Postal Service state abbreviation` AS STRING) AS usps_state_abbreviation,
        CAST(`State FIPS Code` AS STRING) AS state_fips_code,
        CAST(`State GNIS Code` AS STRING) AS state_gnis_code
    FROM v_georef
""")

df_merged = df_rental_mod.join(df_georef_mod, df_rental_mod.state_name == df_georef_mod.usps_state_abbreviation, "inner")

df_merged_filtered = df_merged.filter(df_merged.state_name != "TX")

df_merged_filtered.createOrReplaceTempView("view_car_rental")

spark.sql("""
INSERT INTO TABLE car_rental_db.car_rental_analytics
SELECT * FROM view_car_rental
""")
```

```
hadoop@fda4eaa1811d:~/scripts$ ls
derby.log  ingest_ej1.sh  spark-warehouse    transform_ej1.py  transformation.py
ingest.sh  ingest_ej2.sh  start-services.sh  transform_ej2.py
hadoop@fda4eaa1811d:~/scripts$ ls -l
total 36
-rw-rw-r-- 1 hadoop hadoop  670 Feb 28  2022 derby.log
-rwxrwxr-x 1 hadoop hadoop  258 Feb 28  2022 ingest.sh
-rwxrwxrwx 1 hadoop hadoop  761 Dec  3 01:59 ingest_ej1.sh
-rwxrwxrwx 1 hadoop hadoop  623 Dec 20 04:11 ingest_ej2.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 3195 Dec 18 06:01 transform_ej1.py
-rw-rw-r-- 1 hadoop hadoop 2564 Dec 23 09:32 transform_ej2.py
-rwxrwxrwx 1 hadoop hadoop 1058 May  9  2022 transformation.py
hadoop@fda4eaa1811d:~/scripts$ chmod 777 transform_ej2.py
hadoop@fda4eaa1811d:~/scripts$ ls -l
total 36
-rw-rw-r-- 1 hadoop hadoop  670 Feb 28  2022 derby.log
-rwxrwxr-x 1 hadoop hadoop  258 Feb 28  2022 ingest.sh
-rwxrwxrwx 1 hadoop hadoop  761 Dec  3 01:59 ingest_ej1.sh
-rwxrwxrwx 1 hadoop hadoop  623 Dec 20 04:11 ingest_ej2.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 3195 Dec 18 06:01 transform_ej1.py
-rwxrwxrwx 1 hadoop hadoop 2564 Dec 23 09:32 transform_ej2.py
-rwxrwxrwx 1 hadoop hadoop 1058 May  9  2022 transformation.py
hadoop@fda4eaa1811d:~/scripts$
```

4. Realizar un proceso automático en Airflow que orqueste los pipelines creados en los puntos anteriores. Crear dos tareas:
   a. Un DAG padre que ingente los archivos y luego llame al DAG hijo
   b. Un DAG hijo que procese la información y la cargue en Hive

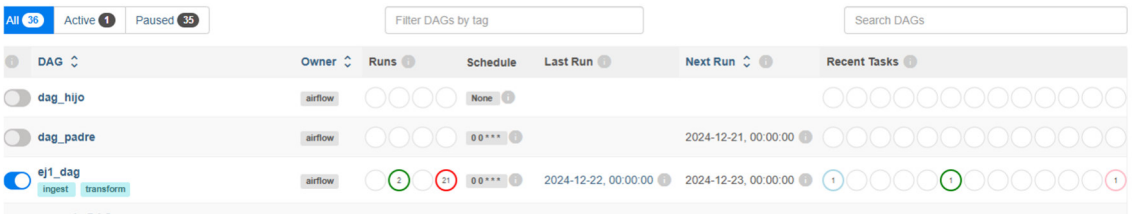
```
 GNU nano 4.8                                  dag_padre.py                                  Modified
from datetime import timedelta
from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.trigger_dagrun import TriggerDagRunOperator
from airflow.operators.dummy import DummyOperator
from airflow.utils.dates import days_ago

# Argumentos
default_args = {
    "owner": "airflow",
    "depends_on_past": False,
    "retries": 1,
    "retry_delay": timedelta(minutes=5),
}

# Creaci      n del DAG
with DAG(
    dag_id="dag_padre",  # Correcci      n del nombre del par      metro
    default_args=default_args,
    schedule_interval="0 0 * * *",  # Ejecuci      n diaria a medianoche
    start_date=days_ago(2),  # Correcci      n de `start_date`
    dagrun_timeout=timedelta(minutes=60),
    catchup=True,
) as dag:
    # Inicio del DAG
    inicia_proceso = DummyOperator(task_id="inicia_proceso")

    # Descarga e ingesta de datos
    download_and_ingest = BashOperator(
        task_id="download_and_ingest",
        bash_command="/usr/bin/sh /home/hadoop/scripts/ingest_final_ej1.sh",
    )
```

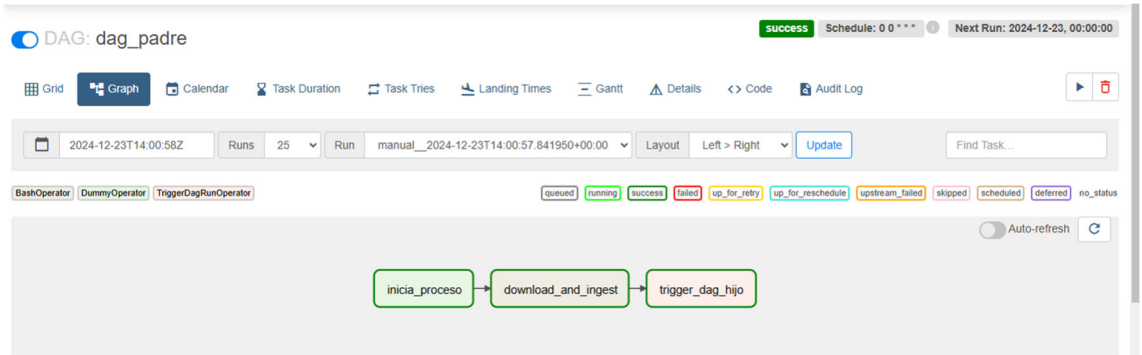
```
    # Dispara el DAG hijo
    trigger_dag_hijo = TriggerDagRunOperator(
        task_id="trigger_dag_hijo",
        trigger_dag_id="dag_hijo",  # Nombre del DAG hijo a disparar
        conf={"execution_date": "{{ ds }}"},  # Par      metro opcional: conf
        reset_dag_run=True,
        wait_for_completion=True,  # Espera la finalizaci      n del DAG hijo
        poke_interval=30,  # Tiempo entre intentos de comprobaci      n
    )

    # Secuencia de tareas
    inicia_proceso >> download_and_ingest >> trigger_dag_hijo
```

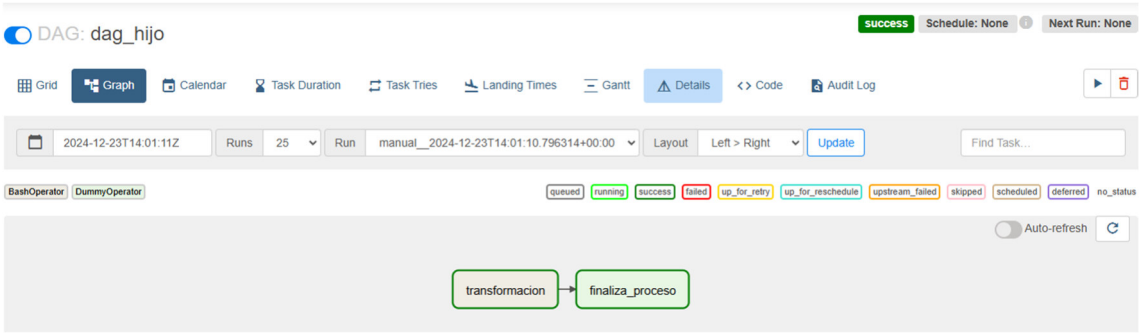
```
 GNU nano 4.8                                  dag_hijo.py                                  Modified
from datetime import timedelta
from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.dummy import DummyOperator
from airflow.utils.dates import days_ago

# Argumentos
default_args = {
    "owner": "airflow",
    "depends_on_past": False,
    "retries": 1,
    "retry_delay": timedelta(minutes=5),
}

# Creaci      n del DAG
with DAG(
    dag_id="dag_hijo",
    default_args=default_args,
    schedule_interval=None,
    start_date=days_ago(2),
    dagrun_timeout=timedelta(minutes=60),
    catchup=False,
) as dag:
    # Tarea 1: Transformaci      n de datos
    transformacion = BashOperator(
        task_id="transformacion",
        bash_command="/home/hadoop/spark/bin/spark-submit --files /home/hadoop/hive/conf/hive-site.xml /home/hadoop/scripts/transformation_>
    )
```

```
    # Tarea 2: Finaliza el proceso
    finaliza_proceso = DummyOperator(task_id="finaliza_proceso")

    # Flujo de tareas
    transformacion >> finaliza_proceso
```

```
hadoop@fda4eaa1811d:~/airflow/dags$ nano dag_padre.py
hadoop@fda4eaa1811d:~/airflow/dags$ nano dag_hijo.py
hadoop@fda4eaa1811d:~/airflow/dags$ chmod 777 dag_padre.py
hadoop@fda4eaa1811d:~/airflow/dags$ chmod 777 dag_hijo.py
hadoop@fda4eaa1811d:~/airflow/dags$ ls -l
total 32
drwxrwxr-x 1 hadoop hadoop 4096 Dec 23 09:46 __pycache__
-rwxrwxrwx 1 hadoop hadoop  982 Dec 23 09:47 dag_hijo.py
-rwxrwxrwx 1 hadoop hadoop 1528 Dec 23 09:45 dag_padre.py
-rwxrwxrwx 1 hadoop hadoop 1344 Dec 16 08:49 ej1_dag.py
-rw-rw-r-- 1 hadoop hadoop 3276 Dec 16 01:41 ej1_transform
-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
-rwxrwxrwx 1 hadoop hadoop 3276 Dec 16 01:42 transform_ej1.py
hadoop@fda4eaa1811d:~/airflow/dags$
```







5. Por medio de consultas SQL al data-warehouse, mostrar:

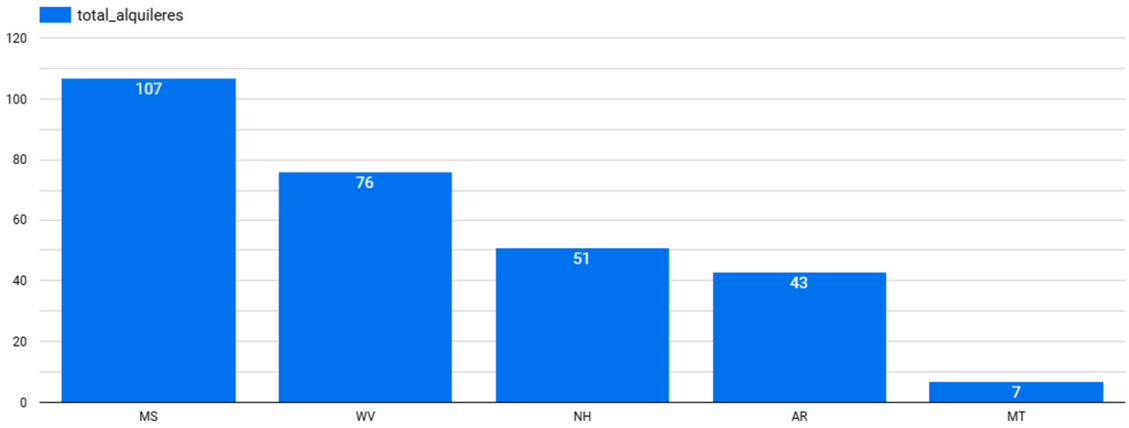
   a. Cantidad de alquileres de autos, teniendo en cuenta sólo los vehículos ecológicos (fuelType hibrido o eléctrico) y con un rating de al menos 4.

```
hive> SELECT SUM(rentertripstaken) AS total_alquileres
    > FROM car_rental_db.car_rental_analytics
    > WHERE fueltype IN ('hybrid', 'electric') AND rating >= 4;
Automatically selecting local only mode for query
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.
e. spark, tez) or using Hive 1.X releases.
Query ID = hadoop_20241230113420_10f87108-c97a-4a44-80e6-791e054b84e5
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2024-12-30 11:34:22,527 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local665290778_0001
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 357916 HDFS Write: 105 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
total_alquileres
26944
Time taken: 2.098 seconds, Fetched: 1 row(s)
hive>
```


   b. los 5 estados con menor cantidad de alquileres (mostrar query y visualización)

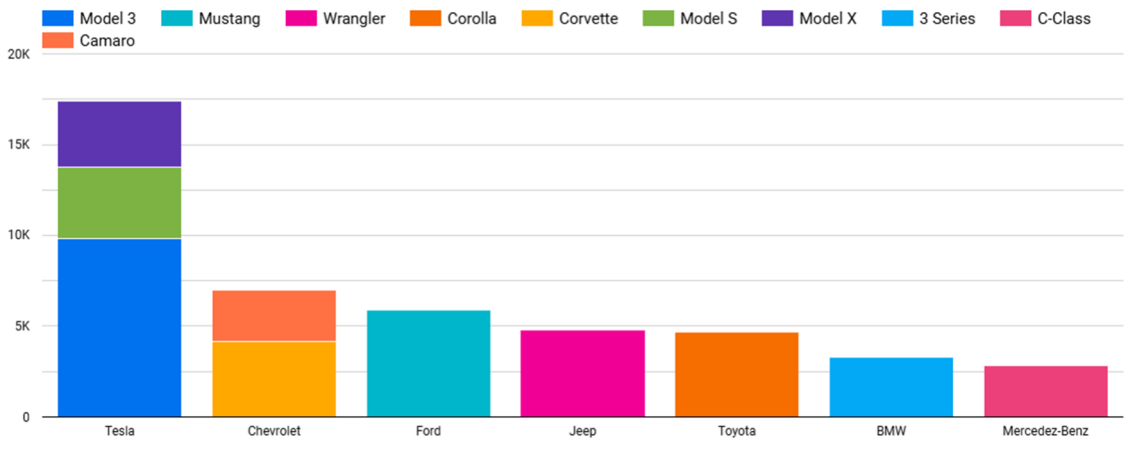
```
hive> SELECT state_name, SUM(rentertripstaken) AS total_alquileres
    > FROM car_rental_db.car_rental_analytics
    > GROUP BY state_name
    > ORDER BY total_alquileres ASC
    > LIMIT 5;
Automatically selecting local only mode for query
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.
e. spark, tez) or using Hive 1.X releases.
Query ID = hadoop_20241230113543_32477476-edc4-4c27-b7d8-08c5a4ded25d
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2024-12-30 11:35:46,432 Stage-1 map = 100%,  reduce = 0%
2024-12-30 11:35:47,436 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local965986135_0002
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Selecting local mode for task: Stage-2
Job running in-process (local Hadoop)
2024-12-30 11:35:49,723 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local1745265115_0003
```

```
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Selecting local mode for task: Stage-2
Job running in-process (local Hadoop)
2024-12-30 11:35:49,723 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local1745265115_0003
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 381654 HDFS Write: 1327 SUCCESS
Stage-Stage-2:  HDFS Read: 391591 HDFS Write: 13347 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
state_name      total_alquileres
MT      7
AR      43
NH      51
WV      76
MS      107
Time taken: 5.968 seconds, Fetched: 5 row(s)
hive>
```



c. los 10 modelos (junto con su marca) de autos más rentados (mostrar query y visualización)

```
hive> SELECT make, model, SUM(rentertripstaken) AS total_alquileres
    > FROM car_rental_db.car_rental_analytics
    > GROUP BY make, model
    > ORDER BY total_alquileres DESC
    > LIMIT 10;
Automatically selecting local only mode for query
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.
e. spark, tez) or using Hive 1.X releases.
Query ID = hadoop_20241230114457_b84864ac-0fba-4f4e-a968-5b15d8b08f88
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2024-12-30 11:44:58,665 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1082059175_0004
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Selecting local mode for task: Stage-2
Job running in-process (local Hadoop)
2024-12-30 11:45:00,364 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local1504777671_0005
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 446298 HDFS Write: 314449 SUCCESS
Stage-Stage-2:  HDFS Read: 490423 HDFS Write: 61320 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
make    model   total_alquileres
Tesla   Model 3 9798
Ford    Mustang 5885
Jeep    Wrangler        4762
Toyota  Corolla 4678
Chevrolet       Corvette        4164
Tesla   Model S 3953
Tesla   Model X 3638
BMW     3 Series        3294
Mercedes-Benz   C-Class 2818
Chevrolet       Camaro  2797
Time taken: 3.004 seconds, Fetched: 10 row(s)
hive>
```

d. Mostrar por año, cuántos alquileres se hicieron, teniendo en cuenta automóviles fabricados desde 2010 a 2015

```
hive> SELECT year, SUM(rentertripstaken) AS total_alquileres
    > FROM car_rental_db.car_rental_analytics
    > WHERE year BETWEEN 2010 AND 2015
    > GROUP BY year
    > ORDER BY year;
Automatically selecting local only mode for query
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.
e. spark, tez) or using Hive 1.X releases.
Query ID = hadoop_20241230115501_ba6aa7dd-40c9-4155-91d8-4e4e344a7208
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2024-12-30 11:55:02,757 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1981124837_0006
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Selecting local mode for task: Stage-2
Job running in-process (local Hadoop)
2024-12-30 11:55:04,854 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local1276939691_0007
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 523498 HDFS Write: 61966 SUCCESS
```

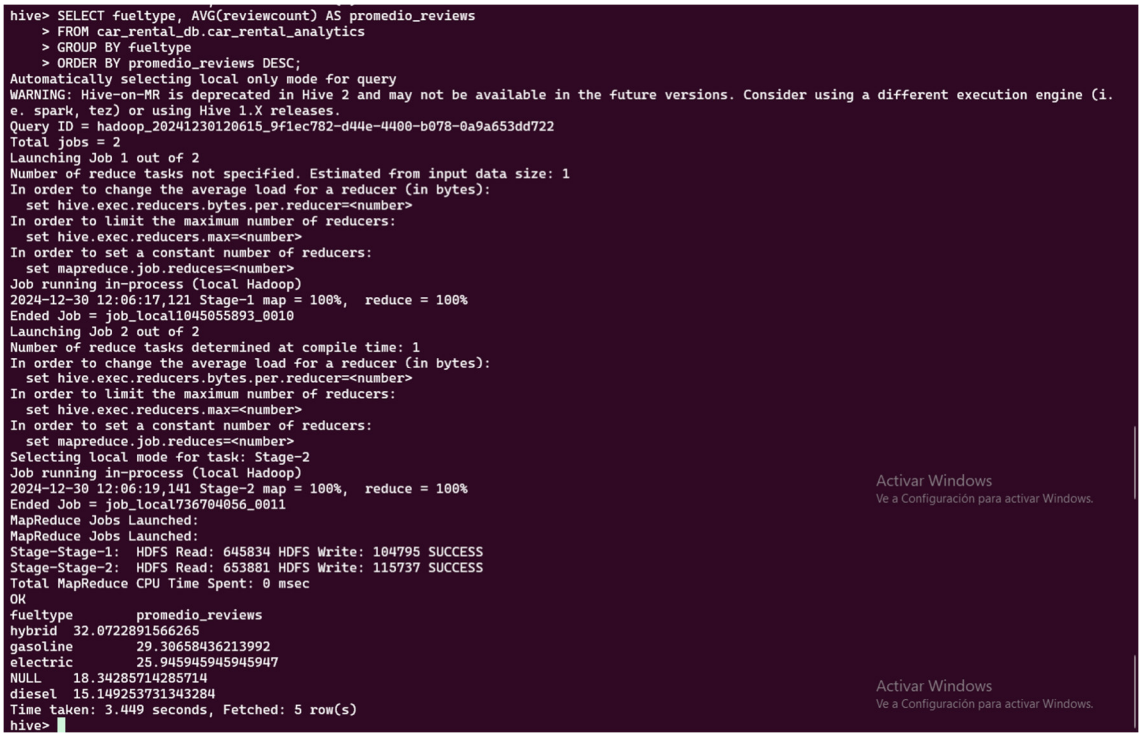


```
Stage-Stage-1:  HDFS Read: 523498 HDFS Write: 61966 SUCCESS
Stage-Stage-2:  HDFS Read: 531456 HDFS Write: 72771 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
year    total_alquileres
2010    6755
2011    8142
2012    10004
2013    12333
2014    15482
2015    18806
Time taken: 3.779 seconds, Fetched: 6 row(s)
hive>
```


e. las 5 ciudades con más alquileres de vehículos ecológicos (fuelType hibrido o electrico)

```
hive> SELECT city, SUM(rentertripstaken) AS total_alquileres
    > FROM car_rental_db.car_rental_analytics
    > WHERE fueltype IN ('hybrid', 'electric')
    > GROUP BY city
    > ORDER BY total_alquileres DESC
    > LIMIT 5;
Automatically selecting local only mode for query
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.
e. spark, tez) or using Hive 1.X releases.
Query ID = hadoop_20241230120348_3b9a0f2d-3bd5-4df7-bcdd-3637b0417924
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2024-12-30 12:03:50,234 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1173739633_0008
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Selecting local mode for task: Stage-2
Job running in-process (local Hadoop)
2024-12-30 12:03:52,232 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local1122123589_0009
MapReduce Jobs Launched:
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Selecting local mode for task: Stage-2
Job running in-process (local Hadoop)
2024-12-30 12:03:52,232 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local1122123589_0009
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 589632 HDFS Write: 83202 SUCCESS
Stage-Stage-2:  HDFS Read: 617716 HDFS Write: 104316 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
city    total_alquileres
San Diego       1795
Las Vegas       1551
Los Angeles     1075
San Francisco   1058
Portland        928
Time taken: 3.455 seconds, Fetched: 5 row(s)
hive>
```

f. el promedio de reviews, segmentando por tipo de combustible



```
hive> SELECT fueltype, AVG(reviewcount) AS promedio_reviews
    > FROM car_rental_db.car_rental_analytics
    > GROUP BY fueltype
    > ORDER BY promedio_reviews DESC;
Automatically selecting local only mode for query
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.
e. spark, tez) or using Hive 1.X releases.
Query ID = hadoop_20241230120615_9f1ec782-d44e-4400-b078-0a9a653dd722
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2024-12-30 12:06:17,121 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1045055893_0010
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Selecting local mode for task: Stage-2
Job running in-process (local Hadoop)
2024-12-30 12:06:19,141 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local736704056_0011
MapReduce Jobs Launched:
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 645834 HDFS Write: 104795 SUCCESS
Stage-Stage-2:  HDFS Read: 653881 HDFS Write: 115737 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
fueltype        promedio_reviews
hybrid  32.0722891566265
gasoline        29.30658436213992
electric        25.945945945945947
NULL    18.342857142857144
diesel  15.149253731343284
Time taken: 3.449 seconds, Fetched: 5 row(s)
hive>
```


6-En base a las consultas, podríamos optar por realizar campañas de difusión y concientización en las ciudades donde no se utilizan o se utilizan menos los vehículos ecológicos.

También podemos reforzar la plaza de autos que mas se alquilaron sabiendo que son de preferencia de los usuarios.

Dada la creciente demanda de alquileres que vemos año a año, podemos analizar aumentar la inversión en los estados donde mas demanda hay e impulsar campañas de publicidad en los estados donde menos alquileres hay.

7-Al igual que en el ejercicio anterior propondría una arquitectura en el entorno cloud de Google (GCP). Principalmente por la disponibilidad online, la capacidad de procesamiento y la robustez.

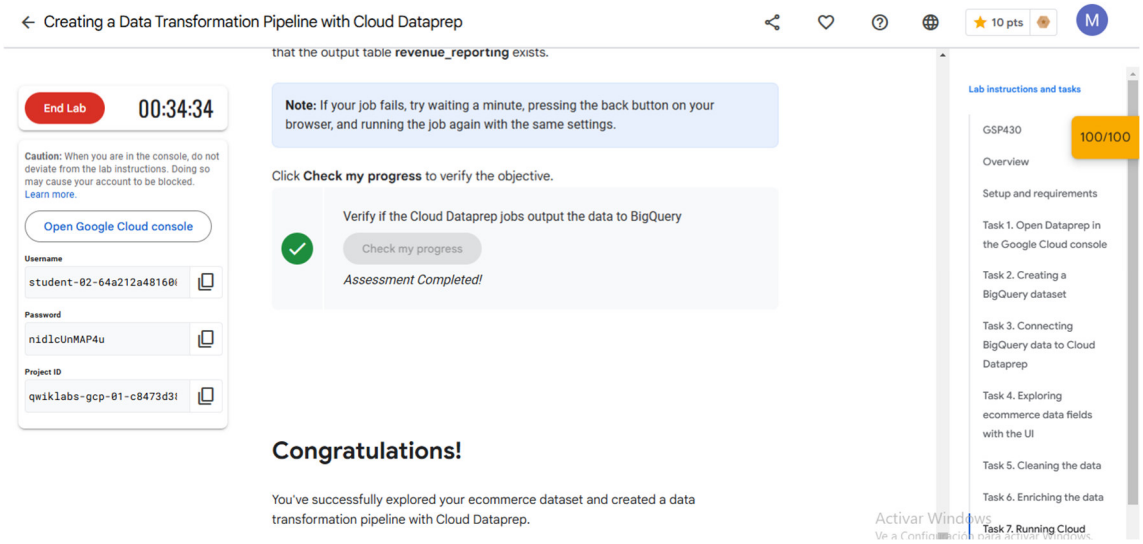
Utilizaría para ingesta Google Cloud Storage, el procesamiento ETL con Dataflow y BigQuery para el almacenamiento. En acaso de automatizar y orquestar el proceso, sería a través de Airflow con Cloud Composer y propondría utilizar Dataprep, para un procesamiento más fino de los datos con fines de enriquecer la base.

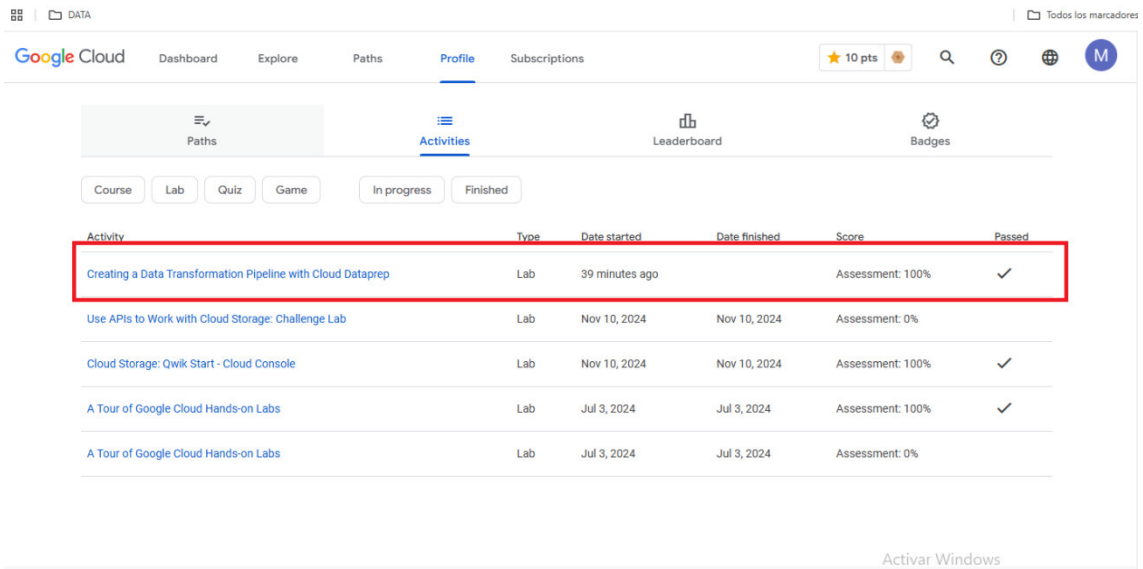
Google Skills Boost - Examen Final

LAB:

Realizar el siguiente LAB, al finalizar pegar un print screen donde se ve su perfil y el progreso final verificado

Título: Creating a Data Transformation Pipeline with Cloud Dataprep
Schedule: 1 hour 15 minutes
Cost: 5 Credits
Link:
https://www.cloudskillsboost.google/focuses/4415?catalog_rank=%7B%22rank%22%3A1%2C%22num_filters%22%3A0%2C%22has_search%22%3Atrue%7D&parent=catalog&search_id=32278924





1. ¿Para qué se utiliza Data Prep?
   Se utiliza para transformar y estructurar los datos sin necesidad de escribir código lo que lo hace una herramienta muy útil para ser utilizada por usuarios menos técnicos.

2. ¿Qué cosas se pueden realizar con DataPrep?

Limpieza de datos, transformación, exploración visual, enriquecimiento y validación de datos, exportación

3. ¿Por qué otra/s herramientas lo podrías reemplazar? ¿Por qué?

Python, ofrece mas flexibilidad y personalización, se puede utilizar con Pandas o Pyspark. Más útil para usuarios técnicos.

Microsoft Power Query por lo que leí en la web es otra herramienta similar, pero para el ecosistema de Azzure.

La elección dependería del caso de uso, presupuesto, nivel técnico del equipo y el ecosistema donde se trabaje.

4. ¿Cuáles son los casos de uso comunes de Data Prep de GCP?

Algunos casos podrían ser:

- Preparación de datos para modelos de aprendizaje automático
- Limpieza y transformación de datos para reportes o dashboards
- Integración de datos de múltiples fuentes
- Creación de pipelines de datos automatizados

5. ¿Cómo se cargan los datos en Data Prep de GCP?

Se pueden cargar de varias fuentes como ser Google Cloud Storage, BigQuery, fuentes externas como API o subida de archivos locales (CSV, JSON, XLS)

6. ¿Qué tipos de datos se pueden preparar en Data Prep de GCP?

Se puede trabajar con:

- Archivos estructurados como CSV, EXCEL o semi estructurados como JSON o XML.

- Bases de datos relacionales externas conectadas o tablas de BigQuery.

- Datos desestructurados, como logs de eventos, texto plano, etc.

7. ¿Qué pasos se pueden seguir para limpiar y transformar datos en Data Prep de GCP?

a) Importar los datos
b) Explorar los datos, para hallar patrones, inconsistencias o nulos
c) Limpiar los datos, limpiar duplicados, gestionar nulos y corregir el formato
d) Transformar los datos, combinar datasets, agregar columnas, realizar operaciones entre columnas
e) Validar los datos, utilizar reglas para comprobar el formato de los datos
f) Exportar los datos

7. ¿Cómo se pueden automatizar tareas de preparación de datos en Data Prep de GCP?

Se pueden definir recipes con los pasos necesarios y las transformaciones, programar Jobs para definir las ejecuciones recurrentes, se puede integrar con Cloud Composer y usar Airflow y se pueden utilizar API's externas.

8. ¿Qué tipos de visualizaciones se pueden crear en Data Prep de GCP?

Histogramas de frecuencias, diagramas de dispersión, de líneas y barras, resúmenes estadísticos, entre otros

9. ¿Cómo se puede garantizar la calidad de los datos en Data Prep de GCP?

Debemos definir reglas de validación, explorar en busca de inconsistencias, automatizar la limpieza para corregir problemas recurrentes, auditar los recipes y hacer validaciones antes de exportar.

Arquitectura:

El gerente de Analitca te pide realizar una arquitectura hecha en GCP que contemple el uso de esta herramienta ya que le parece muy fácil de usar y una interfaz visual que ayuda a sus desarrolladores ya que no necesitan conocer ningún lenguaje de desarrollo.

Esta arquitectura debería contemplar las siguiente etapas:

Ingesta: datos parquet almacenados en un bucket de S3 y datos de una aplicación que guarda sus datos en Cloud SQL.

Procesamiento: filtrar, limpiar y procesar datos provenientes de estas fuentes

Almacenar: almacenar los datos procesados en BigQuery

BI: herramientas para visualizar la información almacenada en el Data Warehouse

ML: Herramienta para construir un modelo de regresión lineal con la información almacenada en el Data Warehouse