

EXAMEN FINAL

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/v/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://data-engineer-edvai-public.s3.amazonaws.com/2021-informe-ministerio.csv>

2022: <https://data-engineer-edvai-public.s3.amazonaws.com/202206-informe-ministerio.csv>

Aeropuertos_detalle:

https://data-engineer-edvai-public.s3.amazonaws.com/aeropuertos_detalle.csv

Comenzaré realizando el **ingest** con el script **ingest_ejercicio_1.sh**, alojado en **/home/hadoop/scripts**. Este script permite descargar los archivos al directorio **/home/hadoop/landing** y posteriormente ingestarlos en **hdfs** en el directorio **/ingest**.

#####

```
# Archivo: ingest_ejercicio_1.sh
```

```
#
```

```
# Descarga e Ingest en HDFS de los archivos:
```

```
# https://data-engineer-edvai-public.s3.amazonaws.com/2021-informe-ministerio.csv
```

```
# https://data-engineer-edvai-public.s3.amazonaws.com/202206-informe-ministerio.csv
```

```
# https://data-engineer-edvai-public.s3.amazonaws.com/aeropuertos_detalle.csv
```

```
#####
```

Punto 1

Descarga datos desde el repositorio al directorio /home/hadoop/landing

```
wget -P /home/hadoop/landing https://data-engineer-edvai-public.s3.amazonaws.com/2021-informe-ministerio.csv
```

```
wget -P /home/hadoop/landing https://data-engineer-edvai-public.s3.amazonaws.com/202206-informe-ministerio.csv
```

```
wget -P /home/hadoop/landing https://data-engineer-edvai-public.s3.amazonaws.com/aeropuertos_detalle.csv
```

Lleva el archivo a HDFS al directorio /ingest

```
hdfs dfs -put /home/hadoop/landing/2021-informe-ministerio.csv /ingest
```

```
hdfs dfs -put /home/hadoop/landing/202206-informe-ministerio.csv /ingest
```

```
hdfs dfs -put /home/hadoop/landing/aeropuertos_detalle.csv /ingest
```

Borra el archivo starwars.csv del directorio /home/hadoop/landing/

```
rm /home/hadoop/landing/2021-informe-ministerio.csv
```

```
rm /home/hadoop/landing/202206-informe-ministerio.csv
```

```
rm /home/hadoop/landing/aeropuertos_detalle.csv
```

Vista desde el **bash**:

```
/home/crbdata/.hushlogin file.
crbdata@RubenBageta:/mnt/c/Users/Carlos Rubén Bageta$ docker exec -it edvai_hadoop bash
root@615cf53bef6c:/# su hadoop
hadoop@615cf53bef6c:/# cd /home/hadoop/scripts
hadoop@615cf53bef6c:/scripts$ ls
Notebook          landing_3.sh      start-services.sh  transform_load_5.py
QueryResult.java  pyspark_jupyter.sh  tl6_run.log        transform_load_6.py
derby.log         query_db_2_hdfs_parquet.sh  transform_load.py  transformation.py
ingest.sh         query_db_2_hdfs_parquet_2.sh  transform_load_2.py  weather_str_transform.py
landing.sh        query_db_2_hdfs_parquet_3.sh  transform_load_3.py  yahoo_weather2kafka.py
landing_2.sh      spark-warehouse    transform_load_4.py
hadoop@615cf53bef6c:/scripts$ cat > ingest_ejercicio_1.sh
#####
# Archivo: ingest_ejercicio_1.sh
#
# Descarga e Ingest en HDFS de los archivos:
# https://data-engineer-edvai-public.s3.amazonaws.com/2021-informe-ministerio.csv
# https://data-engineer-edvai-public.s3.amazonaws.com/202206-informe-ministerio.csv
# https://data-engineer-edvai-public.s3.amazonaws.com/aeropuertos_detalle.csv
#####

# Punto 1

# Descarga datos desde el repositorio al directorio /home/hadoop/landing
wget -P /home/hadoop/landing https://data-engineer-edvai-public.s3.amazonaws.com/2021-informe-ministerio.csv
wget -P /home/hadoop/landing https://data-engineer-edvai-public.s3.amazonaws.com/202206-informe-ministerio.csv
wget -P /home/hadoop/landing https://data-engineer-edvai-public.s3.amazonaws.com/aeropuertos_detalle.csv

# Lleva el archivo a HDFS al directorio /ingest
hdfs dfs -put /home/hadoop/landing/2021-informe-ministerio.csv /ingest
hdfs dfs -put /home/hadoop/landing/202206-informe-ministerio.csv /ingest
hdfs dfs -put /home/hadoop/landing/aeropuertos_detalle.csv /ingest

# Borra el archivo starwars.csv del directorio /home/hadoop/landing/
rm /home/hadoop/landing/2021-informe-ministerio.csv
rm /home/hadoop/landing/202206-informe-ministerio.csv
rm /home/hadoop/landing/aeropuertos_detalle.csv
hadoop@615cf53bef6c:/scripts$
```

A continuación, le asignamos permisos al citado archivo:

```
hadoop@615cf53bef6c:~/scripts$ chmod 777 ingest_ejercicio_1.sh
hadoop@615cf53bef6c:~/scripts$ ls -l
total 144
drwxrwxr-x 3 hadoop hadoop 4096 Sep 29 19:46 Notebook
-rw-rw-r-- 1 hadoop hadoop 14298 Nov 1 15:16 QueryResult.java
-rw-rw-r-- 1 hadoop hadoop 670 Feb 28 2022 derby.log
-rwxrwxr-x 1 hadoop hadoop 272 Sep 22 01:14 ingest.sh
-rwxrwxrwx 1 hadoop hadoop 1393 Nov 15 19:31 ingest_ejercicio_1.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 914 Oct 24 19:25 landing_3.sh
-rwxrwxrwx 1 hadoop hadoop 273 Sep 28 12:14 pyspark_jupyter.sh
-rwxrwxrwx 1 hadoop hadoop 668 Nov 1 15:12 query_db_2_hdfs_parquet.sh
-rwxrwxrwx 1 hadoop hadoop 549 Nov 1 16:00 query_db_2_hdfs_parquet_2.sh
-rwxrwxrwx 1 hadoop hadoop 486 Nov 1 16:31 query_db_2_hdfs_parquet_3.sh
drwxr-xr-x 2 hadoop hadoop 4096 Feb 9 2022 spark-warehouse
-rwxrwxrwx 1 hadoop hadoop 1089 May 9 2022 start-services.sh
-rw-rw-r-- 1 hadoop hadoop 37302 Nov 7 13:41 tl6_run.log
-rwxrwxrwx 1 hadoop hadoop 1964 Oct 19 00:37 transform_load.py
-rwxrwxrwx 1 hadoop hadoop 2897 Oct 19 12:48 transform_load_2.py
-rwxrwxrwx 1 hadoop hadoop 3909 Oct 25 04:35 transform_load_3.py
-rwxrwxrwx 1 hadoop hadoop 1734 Nov 2 23:50 transform_load_4.py
-rwxrwxrwx 1 hadoop hadoop 2619 Nov 3 00:07 transform_load_5.py
-rwxrwxrwx 1 hadoop hadoop 2713 Nov 7 16:27 transform_load_6.py
-rwxrwxrwx 1 hadoop hadoop 1058 May 9 2022 transformation.py
-rwxrwxrwx 1 hadoop hadoop 2068 Oct 31 22:14 weather_str_transform.py
-rwxrwxrwx 1 hadoop hadoop 1747 Nov 1 01:13 yahoo_weather2kafka.py
hadoop@615cf53bef6c:~/scripts$
```

Revisamos que el archivo realiza el **ingest** correctamente:

```
hadoop@615cf53bef6c:~/scripts$ ./ingest_ejercicio_1.sh
--2025-11-15 19:36:28-- https://data-engineer-edvai-public.s3.amazonaws.com/2021-informe-ministerio.csv
Resolving data-engineer-edvai-public.s3.amazonaws.com (data-engineer-edvai-public.s3.amazonaws.com)... 16.15.191.116, 3.5.16.172, 3.5.22.216, ...
Connecting to data-engineer-edvai-public.s3.amazonaws.com (data-engineer-edvai-public.s3.amazonaws.com)|16.15.191.116|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 32322556 (31M) [text/csv]
Saving to: '/home/hadoop/landing/2021-informe-ministerio.csv'

2021-informe-ministerio.csv      100%[=====] 30.82M  157KB/s  in 7m 48s

2025-11-15 19:44:17 (67.5 KB/s) - '/home/hadoop/landing/2021-informe-ministerio.csv' saved [32322556/32322556]

--2025-11-15 19:44:17-- https://data-engineer-edvai-public.s3.amazonaws.com/202206-informe-ministerio.csv
Resolving data-engineer-edvai-public.s3.amazonaws.com (data-engineer-edvai-public.s3.amazonaws.com)... 52.217.228.161, 16.15.185.170, 3.5.22.9, ...
Connecting to data-engineer-edvai-public.s3.amazonaws.com (data-engineer-edvai-public.s3.amazonaws.com)|52.217.228.161|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 22833520 (22M) [text/csv]
Saving to: '/home/hadoop/landing/202206-informe-ministerio.csv'

202206-informe-ministerio.csv   100%[=====] 21.78M  206KB/s  in 2m 52s

2025-11-15 19:47:09 (130 KB/s) - '/home/hadoop/landing/202206-informe-ministerio.csv' saved [22833520/22833520]

--2025-11-15 19:47:09-- https://data-engineer-edvai-public.s3.amazonaws.com/aeropuertos_detalle.csv
Resolving data-engineer-edvai-public.s3.amazonaws.com (data-engineer-edvai-public.s3.amazonaws.com)... 52.216.251.60, 16.15.191.135, 52.217.137.177, ...
Connecting to data-engineer-edvai-public.s3.amazonaws.com (data-engineer-edvai-public.s3.amazonaws.com)|52.216.251.60|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 136007 (133K) [text/csv]
Saving to: '/home/hadoop/landing/aeropuertos_detalle.csv'

aeropuertos_detalle.csv        100%[=====] 132.82K  158KB/s  in 0.8s

2025-11-15 19:47:11 (158 KB/s) - '/home/hadoop/landing/aeropuertos_detalle.csv' saved [136007/136007]

hadoop@615cf53bef6c:~/scripts$ hdfs dfs -ls /ingest
Found 7 items
-rw-r--r-- 1 hadoop supergroup 32322556 2025-11-15 19:47 /ingest/2021-informe-ministerio.csv
-rw-r--r-- 1 hadoop supergroup 22833520 2025-11-15 19:47 /ingest/202206-informe-ministerio.csv
-rw-r--r-- 1 hadoop supergroup 136007 2025-11-15 19:47 /ingest/aeropuertos_detalle.csv
-rw-r--r-- 1 hadoop supergroup 17478 2025-10-25 04:05 /ingest/constructors.csv
-rw-r--r-- 1 hadoop supergroup 94367 2025-10-25 04:05 /ingest/drivers.csv
-rw-r--r-- 1 hadoop supergroup 164344 2025-10-25 04:05 /ingest/races.csv
-rw-r--r-- 1 hadoop supergroup 1721961 2025-10-25 04:05 /ingest/results.csv
hadoop@615cf53bef6c:~/scripts$
```

Se observa que los archivos fueron ingestados correctamente 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)

En primer lugar, creamos la base de datos **aeropuertos** en **Hive**:

```
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> create database aeropuertos;
OK
Time taken: 2.129 seconds
hive> show databases;
OK
aeropuertos
default
fl
northwind_analytics
titanic
tripdata
trips
Time taken: 0.174 seconds, Fetched: 7 row(s)
hive> |
```

Posteriormente, creamos las tablas requeridas (parquet) en la base de datos:

-- TABLA 1: VUELOS (Parquet)

DROP TABLE IF EXISTS aeropuertos.aeropuerto_tabla;

```
CREATE EXTERNAL TABLE aeropuertos.aeropuerto_tabla (
  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
)
STORED AS PARQUET
LOCATION '/tables/external/aeropuertos/aeropuerto_tabla';
```

-- TABLA 2: AEROPUERTOS (Parquet)

DROP TABLE IF EXISTS aeropuertos.aeropuerto_detalles_tabla

```
CREATE EXTERNAL TABLE aeropuertos.aeropuerto_detalles_tabla (
  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
)
STORED AS PARQUET
LOCATION '/tables/external/aeropuertos/aeropuerto_detalle_tabla';

```

Vista desde el **bash**:

```

hive> -- TABLA 1: VUELOS (Parquet)
hive>
    > DROP TABLE IF EXISTS aeropuertos.aeropuerto_tabla;
OK
Time taken: 0.037 seconds
hive> CREATE EXTERNAL TABLE aeropuertos.aeropuerto_tabla (
    >     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
    > )
    > STORED AS PARQUET
    > LOCATION '/tables/external/aeropuertos/aeropuerto_tabla';
OK
Time taken: 0.546 seconds
hive> |

```

```

hive> DROP TABLE IF EXISTS aeropuertos.aeropuerto_detalle_tabla;
OK
Time taken: 0.032 seconds
hive> CREATE EXTERNAL TABLE aeropuertos.aeropuerto_detalle_tabla (
    >     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
    > )
    > STORED AS PARQUET
    > LOCATION '/tables/external/aeropuertos/aeropuerto_detalle_tabla';
OK
Time taken: 0.174 seconds
hive> |

```

Visualizamos que las tablas se crearon correctamente:

```
hive> describe formatted aeropuertos.aeropuerto_tabla
> ;
OK
# col_name          data_type          comment
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

# Detailed Table Information
Database:           aeropuertos
Owner:              hadoop
CreateTime:         Sat Nov 15 21:09:07 ART 2025
LastAccessTime:     UNKNOWN
Retention:          0
Location:           hdfs://172.17.0.2:9000/tables/external/aeropuertos/aeropuerto_tabla
Table Type:         EXTERNAL_TABLE
Table Parameters:
    EXTERNAL              TRUE
    transient_lastDdlTime 1763251747

# Storage Information
SerDe Library:      org.apache.hadoop.hive ql.io.parquet.serde.ParquetHiveSerDe
InputFormat:        org.apache.hadoop.hive ql.io.parquet.MapredParquetInputFormat
OutputFormat:       org.apache.hadoop.hive ql.io.parquet.MapredParquetOutputFormat
Compressed:         No
Num Buckets:        -1
Bucket Columns:     []
Sort Columns:       []
Storage Desc Params:
    serialization.format  1
Time taken: 0.192 seconds, Fetched: 35 row(s)
hive> |
```

```
hive> describe formatted aeropuertos.aeropuerto_detalle_tabla
> ;
OK
# col_name          data_type          comment
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

# Detailed Table Information
Database:           aeropuertos
Owner:              hadoop
CreateTime:         Sat Nov 15 21:12:12 ART 2025
LastAccessTime:     UNKNOWN
Retention:          0
Location:           hdfs://172.17.0.2:9000/tables/external/aeropuertos/aeropuerto_detalle_tabla
Table Type:         EXTERNAL_TABLE
Table Parameters:
    EXTERNAL              TRUE
    transient_lastDdlTime 1763251932

# Storage Information
SerDe Library:      org.apache.hadoop.hive ql.io.parquet.serde.ParquetHiveSerDe
InputFormat:        org.apache.hadoop.hive ql.io.parquet.MapredParquetInputFormat
OutputFormat:       org.apache.hadoop.hive ql.io.parquet.MapredParquetOutputFormat
Compressed:         No
Num Buckets:        -1
Bucket Columns:     []
Sort Columns:       []
Storage Desc Params:
    serialization.format  1
Time taken: 0.096 seconds, Fetched: 46 row(s)
hive> |
```

```
hive> use aeropuertos
> ;
OK
Time taken: 0.054 seconds
hive> show tables;
OK
aeropuerto_detalle_tabla
aeropuerto_tabla
Time taken: 0.066 seconds, Fetched: 2 row(s)
hive> |
```

3. Realizar un proceso automático orquestado por airflow que ingeste los archivos previamente mencionados 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_detalle_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)
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.)

Realizaremos **primero el punto 4**, dado que así podemos hacer las transformaciones con **pyspark** en forma distribuida. EL script **transform_load_7.py**, alojado en **/home/hadoop/scripts** realiza las transformaciones pedidas.

```

1 #####
2 # Archivo: transform_load_7.py
3 #####
4
5 # Importamos librerias
6 from pyspark.sql import SparkSession
7 from pyspark.sql import functions as F
8
9
10 def main():
11     # --- Parámetros de HDFS ---
12     NN_URI = "hdfs://172.17.0.2:9000" # ajusta si tu NameNode/puerto son otros
13     INPUT_INFORME_1 = f"{NN_URI}/ingest/2021-informe-ministerio.csv"
14     INPUT_INFORME_2 = f"{NN_URI}/ingest/202206-informe-ministerio.csv"
15     INPUT_DETALLE_AEROPUERTOS = f"{NN_URI}/ingest/aeropuertos_detalle.csv"
16     OUTPUT_TABLA_1_DW = f"{NN_URI}/tables/external/aeropuertos/aeropuerto_tabla"
17     OUTPUT_TABLA_2_DW = f"{NN_URI}/tables/external/aeropuertos/aeropuerto_detalle_tabla"
18
19     # 1) Crear la sesión de Spark con soporte Hive y FS por defecto en HDFS
20     spark = (
21         SparkSession
22         .builder
23         .appName("transform_load_7")
24         .enableHiveSupport()
25         .config("spark.hadoop.fs.defaultFS", NN_URI)
26         .config("spark.sql.warehouse.dir", OUTPUT_TABLA_1_DW)
27         .config("spark.sql.warehouse.dir", OUTPUT_TABLA_2_DW)
28         .getOrCreate()
29     )
30
31     # 2) Rutas de entrada (csv desde HDFS)
32     df = spark.read.csv(INPUT_INFORME_1, header=True, inferSchema=True, sep=";")
33     df1 = spark.read.csv(INPUT_INFORME_2, header=True, inferSchema=True, sep=";")
34     df2 = spark.read.csv(INPUT_DETALLE_AEROPUERTOS, header=True, inferSchema=True, sep=";")
35
36
37     # 3) Transformaciones (casteo de tipos)
38
39     # Casteo de df y df1 (ademas eliminamos la columna 'Calidad dato' que no es necesaria)
40
41     def castear_vuelos(df): # Funcion para castear los dataframes de vuelos (sin 'Calidad dato')
42         return df.select(
43             F.to_date(F.col("Fecha"), "dd/MM/yyyy").alias("fecha"),
44             F.col("Hora UTC").cast("string").alias("horaUTC"),
45             F.col("Clase de Vuelo (todos los vuelos)").cast("string").alias("clase_de_vuelo"),
46             F.col("Clasificación Vuelo").cast("string").alias("clasificacion_de_vuelo"),
47             F.col("Tipo de Movimiento").cast("string").alias("tipo_de_movimiento"),
48             F.col("Aeropuerto").cast("string").alias("aeropuerto"),
49             F.col("Origen / Destino").cast("string").alias("origen_destino"),
50             F.col("Aerolínea Nombre").cast("string").alias("aerolinea_nombre"),
51             F.col("Aeronave").cast("string").alias("aeronave"),
52             F.col("Pasajeros").cast("int").alias("pasajeros")
53         )
54

```



```

55 # Usamos la función para castear ambos dataframes
56
57 df_cast = castear_vuelos(df)
58 df1_cast = castear_vuelos(df1)
59
60 # Unimos ambos dataframes casteados
61
62 df_union = df_cast.unionByName(df1_cast)
63
64 # Filtramos los vuelos domésticos
65
66 df_domestico = df_union.filter(
67     F.lower(F.col("clasificacion_de_vuelo")) == "domestico"
68 )
69
70 # Rellenamos los valores nulos en la columna 'pasajeros' con 0
71
72 df_load = df_domestico.fillna({"pasajeros": 0})
73
74 # Casteo de df2 (aeropuertos detalle) No figura la columna 'inhab' ni 'fir' en la tabla destino, por lo que no se incluyen
75
76 df2_cast = df2.select(
77     # local -> aeropuerto
78     F.col("local").cast("string").alias("aeropuerto"),
79     F.col("oaci").cast("string").alias("oac"),
80     F.col("iata").cast("string").alias("iata"),
81     F.col("tipo").cast("string").alias("tipo"),
82     F.col("denominacion").cast("string").alias("denominacion"),
83     F.col("coordenadas").cast("string").alias("coordenadas"),
84     F.col("latitud").cast("string").alias("latitud"),
85     F.col("longitud").cast("string").alias("longitud"),
86     F.col("elev").cast("float").alias("elev"),
87     F.col("uom_elev").cast("string").alias("uom_elev"),
88     F.col("ref").cast("string").alias("ref"),
89     F.col("distancia_ref").cast("float").alias("distancia_ref"),
90     F.col("direccion_ref").cast("string").alias("direccion_ref"),
91     F.col("condicion").cast("string").alias("condicion"),
92     F.col("control").cast("string").alias("control"),
93     F.col("region").cast("string").alias("region"),
94     F.col("uso").cast("string").alias("uso"),
95     F.col("trafico").cast("string").alias("trafico"),
96     F.col("sna").cast("string").alias("sna"),
97     F.col("concesionado").cast("string").alias("concesionado"),
98     F.col("provincia").cast("string").alias("provincia"),
99
100 )
101
102 # Rellenamos los valores nulos en la columna 'distancia_ref' con 0
103
104 df2_load = df2_cast.fillna({"distancia_ref": 0})
105
106
107 # (opcional) métricas rápidas al log
108 print(f"[INFO] filas vuelos : {df_load.count()}")
109 print(f"[INFO] filas aeropuertos detalles : {df2_load.count()}")
110
111 # 5) Load (Parquet a HDFS)
112 df_load.write.mode("overwrite").parquet(OUTPUT_TABLA_1_DW)
113 df2_load.write.mode("overwrite").parquet(OUTPUT_TABLA_2_DW)
114
115 spark.stop()
116
117 if __name__ == "__main__":
118     main()

```

Vista desde el **bash**:

```

hadoop@615cf53bef6c:~/scripts$ ls
Netscok      ingest.sh      landing_2.sh      query_db_2_hdfs_parquet.sh      spark-warehouse      transform_load.py      transform_load_4.py      transformation.py
QueryResult.java  ingest_ejercicio_1.sh  landing_3.sh      query_db_2_hdfs_parquet_2.sh      start-services.sh      transform_load_2.py      transform_load_5.py      weather_str_transform.py
derby.log      landing.sh      pyspark_jupyter.sh      query_db_2_hdfs_parquet_3.sh      tl6_run.log      transform_load_3.py      transform_load_6.py      yahoo_weather2kafka.py
hadoop@615cf53bef6c:~/scripts$ cat > transform_load_7.py
#####
# Archivo: transform_load_7.py
#####

# Importamos librerias
from pyspark.sql import SparkSession
from pyspark.sql import functions as F

def main():
    # --- Parámetros de HDFS ---
    MN_URI = "hdfs://172.17.0.2:9080" # ajusta si tu NameNode/puerto son otros
    INPUT_INFORME_1 = F*(MN_URI)/ingest/2021-informa-ministerio.csv"
    INPUT_INFORME_2 = F*(MN_URI)/ingest/2022086-informa-ministerio.csv"
    INPUT_DETALLE_AEROPUERTOS = F*(MN_URI)/ingest/aeropuertos_detalle.csv"
    OUTPUT_TABLA_1_DW = F*(MN_URI)/tables/external/aeropuertos/aeropuerto_tabla"
    OUTPUT_TABLA_2_DW = F*(MN_URI)/tables/external/aeropuertos/aeropuerto_detalle_tabla"

    # 1) Crear la sesión de Spark con soporte Hive y FS por defecto en HDFS
    spark = (
        SparkSession
        .builder
        .appName("transform_load_7")
        .enableHiveSupport()
        .config("spark.hadoop.fs.defaultFS", MN_URI)
        .config("spark.sql.warehouse.dir", OUTPUT_TABLA_1_DW)
        .config("spark.sql.warehouse.dir", OUTPUT_TABLA_2_DW)
        .getOrCreate()
    )

    # 2) Rutas de entrada (csv desde HDFS)
    df = spark.read.csv(INPUT_INFORME_1, header=True, inferSchema=True, sep=";")
    df1 = spark.read.csv(INPUT_INFORME_2, header=True, inferSchema=True, sep=";")
    df2 = spark.read.csv(INPUT_DETALLE_AEROPUERTOS, header=True, inferSchema=True, sep=";")

    # 3) Transformaciones (casteo de tipos)
    # Casteo de df y df1 (ademas eliminamos la columna 'Calidad dato' que no es necesaria)

    def castear_vuelos(df): # Funcion para castear los dataframes de vuelos (sin 'Calidad dato')
        return df.select(
            F.to_date(F.col("Fecha"), "dd/MM/yyyy").alias("fecha"),
            F.col("hora UTC").cast("string").alias("horaUTC"),
            F.col("clase de Vuelo (todos los vuelos)").cast("string").alias("clase de vuelo"),
            F.col("clasificación Vuelo").cast("string").alias("clasificación de vuelo"),
            F.col("tipo de Movimiento").cast("string").alias("tipo de movimiento"),
            F.col("aeropuerto").cast("string").alias("aeropuerto"),
            F.col("Origen / Destino").cast("string").alias("origen_destino"),
            F.col("Aerolinea Nombre").cast("string").alias("aerolinea_nombre"),
            F.col("Aeromane").cast("string").alias("aeromane"),
            F.col("Pasajeros").cast("int").alias("pasajeros"),
        )

    # Usamos la función para castear ambos dataframes
    df_cast = castear_vuelos(df)
    df1_cast = castear_vuelos(df1)

```

```

# Usamos la función para castear ambos dataframes

df_cast = castear_vuelos(df)
df1_cast = castear_vuelos(df1)

# Unimos ambos dataframes casteados

df_union = df_cast.unionByName(df1_cast)

# Filtramos los vuelos domésticos
df_domestico = df_union.filter(
    F.lower(F.col("clasificación de vuelo")) == "domestico"
)

# Rellenamos los valores nulos en la columna 'pasajeros' con 0

df_load = df_domestico.fillna({"pasajeros": 0})

# Casteo de df2 (aeropuertos detalle) No figura la columna 'inhab' ni 'fir' en la tabla destino, por lo que no se incluyen

df2_cast = df2.select(
    # local -> aeropuerto
    F.col("local").cast("string").alias("aeropuerto"),
    F.col("oaci").cast("string").alias("oac"),
    F.col("iata").cast("string").alias("iata"),
    F.col("tipo").cast("string").alias("tipo"),
    F.col("denominación").cast("string").alias("denominación"),
    F.col("coordenadas").cast("string").alias("coordenadas"),
    F.col("latitud").cast("string").alias("latitud"),
    F.col("longitud").cast("string").alias("longitud"),
    F.col("elev").cast("float").alias("elev"),
    F.col("uom_elev").cast("string").alias("uom_elev"),
    F.col("ref").cast("string").alias("ref"),
    F.col("distancia_ref").cast("float").alias("distancia_ref"),
    F.col("direccion_ref").cast("string").alias("direccion_ref"),
    F.col("condicion").cast("string").alias("condicion"),
    F.col("control").cast("string").alias("control"),
    F.col("region").cast("string").alias("region"),
    F.col("uso").cast("string").alias("uso"),
    F.col("trafico").cast("string").alias("trafico"),
    F.col("sna").cast("string").alias("sna"),
    F.col("concesionado").cast("string").alias("concesionado"),
    F.col("provincia").cast("string").alias("provincia"),
)

# Rellenamos los valores nulos en la columna 'distancia_ref' con 0

df2_load = df2_cast.fillna({"distancia_ref": 0})

# (opcional) métricas rápidas al log
print(f"[INFO] filas vuelos      : {df_load.count()}")
print(f"[INFO] filas aeropuertos detalles : {df2_load.count()}")

# 5) Load (Parquet a HDFS)
df_load.write.mode("overwrite").parquet(OUTPUT_TABLA_1_DW)
df2_load.write.mode("overwrite").parquet(OUTPUT_TABLA_2_DW)

spark.stop()

if __name__ == "__main__":
    main()
hadoop@615cf53bef6c:~/scripts$ |

```

Asignamos permisos:

```
hadoop@615cf53bef6c:~/scripts$ chmod 777 transform_load_7.py
hadoop@615cf53bef6c:~/scripts$ ls -l
total 152
drwxrwxr-x 3 hadoop hadoop 4096 Sep 29 19:46 Notebook
-rw-rw-r-- 1 hadoop hadoop 14298 Nov  1 15:16 QueryResult.java
-rw-rw-r-- 1 hadoop hadoop  670 Feb 28  2022 derby.log
-rwxrwxr-x 1 hadoop hadoop  272 Sep 22 01:14 ingest.sh
-rwxrwxrwx 1 hadoop hadoop 1393 Nov 15 19:31 ingest_ejercicio_1.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  914 Oct 24 19:25 landing_3.sh
-rwxrwxrwx 1 hadoop hadoop  273 Sep 28 12:14 pyspark_jupyter.sh
-rwxrwxrwx 1 hadoop hadoop  668 Nov  1 15:12 query_db_2_hdfs_parquet.sh
-rwxrwxrwx 1 hadoop hadoop  549 Nov  1 16:00 query_db_2_hdfs_parquet_2.sh
-rwxrwxrwx 1 hadoop hadoop  486 Nov  1 16:31 query_db_2_hdfs_parquet_3.sh
drwxr-xr-x 2 hadoop hadoop 4096 Feb  9  2022 spark-warehouse
-rwxrwxrwx 1 hadoop hadoop 1089 May  9  2022 start-services.sh
-rw-rw-r-- 1 hadoop hadoop 37302 Nov  7 13:41 tl6_run.log
-rwxrwxrwx 1 hadoop hadoop 1964 Oct 19 00:37 transform_load.py
-rwxrwxrwx 1 hadoop hadoop 2897 Oct 19 12:48 transform_load_2.py
-rwxrwxrwx 1 hadoop hadoop 3909 Oct 25 04:35 transform_load_3.py
-rwxrwxrwx 1 hadoop hadoop 1734 Nov  2 23:50 transform_load_4.py
-rwxrwxrwx 1 hadoop hadoop 2619 Nov  3 00:07 transform_load_5.py
-rwxrwxrwx 1 hadoop hadoop 2713 Nov  7 16:27 transform_load_6.py
-rwxrwxrwx 1 hadoop hadoop 5065 Nov 16 01:36 transform_load_7.py
-rwxrwxrwx 1 hadoop hadoop 1058 May  9  2022 transformation.py
-rwxrwxrwx 1 hadoop hadoop 2068 Oct 31 22:14 weather_str_transform.py
-rwxrwxrwx 1 hadoop hadoop 1747 Nov  1 01:13 yahoo_weather2kafka.py
hadoop@615cf53bef6c:~/scripts$
```

Revisamos que el script funcione correctamente:

```
hadoop@615cf53bef6c:~/scripts$ /home/hadoop/spark/bin/spark-submit --master local[*] \
> --conf spark.sql.warehouse.dir=/user/hive/warehouse \
> --conf spark.sql.catalogImplementation=hive \
> --files /home --files /home/hadoop/hive/conf/hive-site.xml \
> /home/hadoop/scripts/transform_load_7.py 2>&1 | tee /home/hadoop/scripts/tl7_run.log
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/home/hadoop/spark/jars/spark-unsafe-2.12-3.2.0.jar) to constructor java.nio.DirectByteBuffer
(Long,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
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
2025-11-16 01:47:18,495 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2025-11-16 01:47:19,737 INFO spark.SparkContext: Running Spark version 3.2.0
2025-11-16 01:47:19,791 INFO resource.ResourceUtils: =====
2025-11-16 01:47:19,792 INFO resource.ResourceUtils: No custom resources configured for spark.driver.
2025-11-16 01:47:19,792 INFO resource.ResourceUtils: =====
2025-11-16 01:47:19,793 INFO spark.SparkContext: Submitted application: transform_load_7
2025-11-16 01:47:19,863 INFO resource.ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script: , vendor: , memor
y -> name: memory, amount: 1024, script: , offHeap -> name: offHeap, amount: 0, script: , vendor: ), task resources: Map(cpus -> name: cpus, amount: 1.0)
2025-11-16 01:47:19,865 INFO resource.ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script: , vendor: , memor
y -> name: memory, amount: 1024, script: , offHeap -> name: offHeap, amount: 0, script: , vendor: ), task resources: Map(cpus -> name: cpus, amount: 1.0)
2025-11-16 01:47:39,513 INFO datasources.FileScanRDD: Reading File path: hdfs://172.17.0.2:9000/ingest/aeropuertos_detalle.csv, range: 0-136007, partition values: [empty
row]
2025-11-16 01:47:39,531 INFO codegen.CodeGenerator: Code generated in 13.667123 ms
2025-11-16 01:47:39,541 INFO input.LineRecordReader: Found UTF-8 BOM and skipped it
2025-11-16 01:47:39,600 INFO storage.BlockManagerInfo: Removed broadcast_21_piece0 on 615cf53bef6c:33741 in memory (size: 130.1 KiB, free: 434.1 MiB)
2025-11-16 01:47:39,663 INFO output.FileOutputCommitter: Saved output of task 'attempt_202511160147396429773219028097960_0013_m_000000_49' to hdfs://172.17.0.2:9000/table
s/external/aeropuertos/aeropuerto_detalle_tabla/-temporary/0/task_202511160147396429773219028097960_0013_m_000000
2025-11-16 01:47:39,664 INFO mapred.SparkHadoopMapRedUtil: attempt_202511160147396429773219028097960_0013_m_000000_49: Committed
2025-11-16 01:47:39,665 INFO executor.Executor: Finished task 0.0 in stage 13.0 (TID 49). 2558 bytes result sent to driver
2025-11-16 01:47:39,667 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 13.0 (TID 49) in 227 ms on 615cf53bef6c (executor driver) (1/1)
2025-11-16 01:47:39,667 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 13.0, whose tasks have all completed, from pool
2025-11-16 01:47:39,668 INFO scheduler.DAGScheduler: ResultStage 13 (parquet at NativeMethodAccessorImpl.java:0) finished in 0.258 s
2025-11-16 01:47:39,668 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 13: Stage finished
2025-11-16 01:47:39,669 INFO scheduler.DAGScheduler: Job 11 is finished. Cancelling potential speculative or zombie tasks for this job
2025-11-16 01:47:39,669 INFO scheduler.DAGScheduler: Job 11 finished: parquet at NativeMethodAccessorImpl.java:0, took 0.262936 s
2025-11-16 01:47:39,670 INFO datasources.FileFormatWriter: Start to commit write Job 7729f62d-1f97-4b84-8232-b362473a585b.
2025-11-16 01:47:39,694 INFO datasources.FileFormatWriter: Write Job 7729f62d-1f97-4b84-8232-b362473a585b committed. Elapsed time: 23 ms.
2025-11-16 01:47:39,694 INFO datasources.FileFormatWriter: Finished processing stats for write Job 7729f62d-1f97-4b84-8232-b362473a585b.
2025-11-16 01:47:39,710 INFO server.AbstractConnector: Stopped Spark@4789a95(HTTP/1.1, (http/1.1)){0.0.0.0:4041}
2025-11-16 01:47:39,712 INFO ui.SparkUI: Stopped Spark web UI at http://615cf53bef6c:4041
2025-11-16 01:47:39,739 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
2025-11-16 01:47:39,769 INFO memory.MemoryStore: MemoryStore cleared
2025-11-16 01:47:39,770 INFO storage.BlockManager: BlockManager stopped
2025-11-16 01:47:39,775 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
2025-11-16 01:47:39,781 INFO scheduler.OutputCommitCoordinator: OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
2025-11-16 01:47:39,804 INFO spark.SparkContext: Successfully stopped SparkContext
2025-11-16 01:47:40,135 INFO util.ShutdownHookManager: Shutdown hook called
2025-11-16 01:47:40,135 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-23bb4877-09e9-4349-8c7d-830996c689eb
2025-11-16 01:47:40,142 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-62e33a41-ea1f-4336-b25a-a5e5db40bad6/pyspark-5bf8ea6c-ae9-4186-9793-8643f6ae2a19
2025-11-16 01:47:40,149 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-62e33a41-ea1f-4336-b25a-a5e5db40bad6
hadoop@615cf53bef6c:~/scripts$
```

Procederemos con el **punto 5**, a mostrar los tipos de los campos de los dataframes obtenidos.

```
>>> df_load.printSchema()
root
|-- fecha: date (nullable = true)
|-- horaUTC: string (nullable = true)
|-- clase_de_vuelo: string (nullable = true)
|-- clasificacion_de_vuelo: string (nullable = true)
|-- tipo_de_movimiento: string (nullable = true)
|-- aeropuerto: string (nullable = true)
|-- origen_destino: string (nullable = true)
|-- aerolinea_nombre: string (nullable = true)
|-- aeronave: string (nullable = true)
|-- pasajeros: integer (nullable = false)
```

```
>>> df2_load.printSchema()
root
|-- aeropuerto: string (nullable = true)
|-- oac: string (nullable = true)
|-- iata: string (nullable = true)
|-- tipo: string (nullable = true)
|-- denominacion: string (nullable = true)
|-- coordenadas: string (nullable = true)
|-- latitud: string (nullable = true)
|-- longitud: string (nullable = true)
|-- elev: float (nullable = true)
|-- uom_elev: string (nullable = true)
|-- ref: string (nullable = true)
|-- distancia_ref: float (nullable = false)
|-- direccion_ref: string (nullable = true)
|-- condicion: string (nullable = true)
|-- control: string (nullable = true)
|-- region: string (nullable = true)
|-- uso: string (nullable = true)
|-- trafico: string (nullable = true)
|-- sna: string (nullable = true)
|-- concesionado: string (nullable = true)
|-- provincia: string (nullable = true)
```

Finalmente avanzaremos en el punto 3, armaremos el piplane que realiza la transformación y carga de los datos en el warehouse.

Creemos el DAG para Airflow:

```
Ejercicio_1 > DAG_ejercicio_1_trabajo_final.py > ...
1 #####
2 #
3 # DAG creado para el ejercicio 1 Trabajo Final
4 # /home/hadoop/airflow/dags/DAG_ejercicio_1_trabajo_final.py
5 #
6 #####
7
8 # Importamos librerias
9
10 from datetime import timedelta
11 from airflow import DAG
12 from airflow.operators.bash import BashOperator
13 from airflow.operators.dummy import DummyOperator
14 from airflow.utils.dates import days_ago
15 from airflow.utils.task_group import TaskGroup
16
17
18 # Definimos los argumentos por defecto del DAG
19 args = {
20     "owner": "airflow",
21 }
22
23 with DAG(
24     dag_id="ingest-transform-7",
25     default_args=args,
26     schedule_interval="0 0 * * *",
27     start_date=days_ago(1),
28     catchup=False,
29     dagrun_timeout=timedelta(minutes=60),
30     tags=["ingest", "transform"],
31 ) as dag:
32
33     inicio_proceso = DummyOperator(task_id="inicio_proceso")
34
35     finaliza_proceso = DummyOperator(task_id="finaliza_proceso")
36
37     # Tareas Ingest - Transform - Load (sin cambios)
38
39     with TaskGroup("ETL", tooltip="Tareas de Extracción, Transformacion y Carga de Datos") as process_group:
40         products_sold = BashOperator(
41             task_id='ETL_aeropuertos',
42             bash_command="""
43                 set -e
44                 /home/hadoop/spark/bin/spark-submit \
45                 --master local[*] \
46                 --deploy-mode client \
47                 --files /home/hadoop/hive/conf/hive-site.xml \
48                 /home/hadoop/scripts/transform_load_7.py
49             """,
50         )
51
52     # Inicio del flujo de tareas ETL
53     inicio_proceso >> process_group
54
55     # Finalizacion del proceso de tareas
56     process_group >> finaliza_proceso
57
```

En **bash** creamos el archivo en **/home/hadoop/airflow/dags/**:

```

This message is shown once a day. To disable it please create the
/home/crbdata/.hushlogin file.
crbdata@RubenBageta:/mnt/c/Users/Carlos Rubén Bageta$ docker exec -it edvai_hadoop bash
root@615cf53bef6c:/# su hadoop
hadoop@615cf53bef6c:/# cd /home/hadoop/airflow/dags/
hadoop@615cf53bef6c:~/airflow/dags$ cat > DAG_ejercicio_1_trabajo_final.py
#####
#
# DAG creado para el ejercicio 1 Trabajo Final
# /home/hadoop/airflow/dags/DAG_ejercicio_1_trabajo_final.py
#
#####

# Importamos librerias

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
from airflow.utils.task_group import TaskGroup

# Definimos los argumentos por defecto del DAG
args = {
    "owner": "airflow",
}

with DAG(
    dag_id="ingest-transform-7",
    default_args=args,
    schedule_interval="@ 0 * * *",
    start_date=days_ago(1),
    catchup=False,
    dagrun_timeout=timedelta(minutes=60),
    tags=["ingest", "transform"],
) as dag:

    inicio_proceso = DummyOperator(task_id="inicio_proceso")

    finaliza_proceso = DummyOperator(task_id="finaliza_proceso")

    # Tareas Ingest - Transform - Load (sin cambios)

    with TaskGroup("ETL", tooltip="Tareas de Extracción, Transformacion y Carga de Datos") as process_group:
        products_sold = BashOperator(
            task_id='ETL_aeropuertos',
            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_7.py
            """,
        )

    # Inicio del flujo de tareas ETL
    inicio_proceso >> process_group

    # Finalizacion del proceso de tareas
    process_group >> finaliza_proceso

hadoop@615cf53bef6c:~/airflow/dags$

```

Asignamos permisos al DAG:


```

hadoop@615cf53bef6c:~/airflow/dags$ chmod 777 DAG_ejercicio_1_trabajo_final.py
hadoop@615cf53bef6c:~/airflow/dags$ ls -l
total 40
-rwxrwxrwx 1 hadoop hadoop 1679 Nov 16 12:35 DAG_ejercicio_1_trabajo_final.py
-rw-rw-r-- 1 hadoop hadoop 1663 Oct 19 12:59 DAG_ejercicio_5.py
-rwxrwxrwx 1 hadoop hadoop 2893 Oct 25 01:49 DAG_ejercicio_5_clase_8.py
-rwxrwxrwx 1 hadoop hadoop 1742 Nov 7 15:16 DAG_ejercicio_7_clase_10.py
-rwxrwxrwx 1 hadoop hadoop 4009 Nov 2 23:53 DAG_ejercicio_7_clase_9.py
-rw-rw-r-- 1 hadoop hadoop 1496 Oct 31 21:15 DAG_str_weather.py
-rwxrwxrwx 1 hadoop hadoop 542 Nov 1 00:14 DAG_weather2kafka.py
drwxrwxrwx 2 hadoop hadoop 4096 Nov 16 12:38 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
hadoop@615cf53bef6c:~/airflow/dags$

```

Corremos el DAG:

The screenshot shows the Apache Airflow web interface. At the top, there's a navigation bar with links for DAGs, Security, Browse, Admin, and Docs. Below this, a status bar indicates that the DAG 'ingest-transform-7' was triggered and should start any moment now. The main section shows the DAG 'ingest-transform-7' in a 'running' state. The DAG is a simple linear flow: 'inicio_proceso' -> 'ETL' -> 'finaliza_proceso'. The interface includes a top navigation bar, a status bar indicating the DAG is triggered, and a task list at the bottom.

Verificamos ahora que se hizo correctamente el Load en las tablas correspondientes (lo hacemos desde **DBeaver**). Para esto armamos una conexión con **Hive** para leer el contenido de las tablas.

Tabla aeropuerto_tabla

id	fecha	hora_salida	clase_vuelo	clasificacion_vuelo	tipo_movimiento	aeropuerto	origen_destino	compania_nombre	aeronave	pasajeros
1	2022-06-30	00:02	Regular	Doméstico	Aterrizaje	BAR	AER	JETSMART AIRLINES S.A.	AIB-A320-232	0
2	2022-06-30	00:07	No Regular	Doméstico	Aterrizaje	FDO	CBA	EMB-EMB-300	0	0
3	2022-06-30	00:08	Regular	Doméstico	Aterrizaje	ROS	FMA	AEROLINEAS ARGENTINAS SA	EMB-ER190100GW	48
4	2022-06-30	00:09	Vuelo Privado con Matricula Nacional	Doméstico	Despegue	FDO	FDO	TEA-P2002 SIERRA MKI	0	0
5	2022-06-30	00:10	Vuelo Privado con Matricula Nacional	Doméstico	Despegue	OSA	FDO	BE-B-200	0	0
6	2022-06-30	00:12	Regular	Doméstico	Despegue	JULI	EZE	AEROLINEAS ARGENTINAS SA	BO-8737-800	0
7	2022-06-30	00:15	Regular	Doméstico	Aterrizaje	TUC	AER	AEROLINEAS ARGENTINAS SA	BO-8737-8LP	0
8	2022-06-30	00:15	Vuelo Privado con Matricula Nacional	Doméstico	Despegue	ROS	ROS	0	PAJ-PA-A-38-112	0
9	2022-06-30	00:17	Regular	Doméstico	Aterrizaje	BAR	AER	FB LINEAS AEREAS - FLYBOND	BO-737-802B	95
10	2022-06-30	00:17	No Regular	Doméstico	Aterrizaje	FDO	TUC	0	FA-SA-226-TC	1
11	2022-06-30	00:18	Regular	Doméstico	Aterrizaje	AER	ROS	AEROLINEAS ARGENTINAS SA	BO-8737-800	0
12	2022-06-30	00:21	Regular	Doméstico	Aterrizaje	AER	DRV	AEROLINEAS ARGENTINAS SA	BO-8737-8BK	71
13	2022-06-30	00:24	Regular	Doméstico	Aterrizaje	DOZ	AER	AEROLINEAS ARGENTINAS SA	BO-737-8BK	0
14	2022-06-30	00:24	Regular	Doméstico	Aterrizaje	IGU	AER	AEROLINEAS ARGENTINAS SA	BO-737-84K	86
15	2022-06-30	00:26	Regular	Doméstico	Aterrizaje	EZE	IGU	JETSMART AIRLINES S.A.	AIB-A320-232	93
16	2022-06-30	00:27	Regular	Doméstico	Aterrizaje	AER	SAL	JETSMART AIRLINES S.A.	AIB-A320-232	73
17	2022-06-30	00:32	Regular	Doméstico	Aterrizaje	SHV	EZE	AEROLINEAS ARGENTINAS SA	EMB-ER190100GW	0
18	2022-06-30	00:33	Regular	Doméstico	Aterrizaje	EZE	IGU	AEROLINEAS ARGENTINAS SA	BO-8737-889	77
19	2022-06-30	00:41	Regular	Doméstico	Despegue	SAL	EZE	FB LINEAS AEREAS - FLYBOND	BO-737-8AL	0
20	2022-06-30	00:42	Regular	Doméstico	Aterrizaje	AER	BAR	AEROLINEAS ARGENTINAS SA	BO-8-737-8TD	0
21	2022-06-30	00:44	Regular	Doméstico	Despegue	CBA	AER	JETSMART AIRLINES S.A.	AIB-A320-232	86
22	2022-06-30	00:47	Regular	Doméstico	Despegue	ROS	AER	AEROLINEAS ARGENTINAS SA	EMB-ER190100GW	11
23	2022-06-30	00:48	Regular	Doméstico	Despegue	ECA	EZE	AEROLINEAS ARGENTINAS SA	BO-8737-8MB	0
24	2022-06-30	00:49	Regular	Doméstico	Aterrizaje	CBA	AER	AEROLINEAS ARGENTINAS SA	BO-737-84K	0
25	2022-06-30	00:49	Vuelo Privado con Matricula Nacional	Doméstico	Aterrizaje	FDO	FDO	0	TEA-P2002 SIERRA MKI	0
26	2022-06-30	00:50	Regular	Doméstico	Despegue	ROS	AER	AEROLINEAS ARGENTINAS SA	BO-737-85H	0
27	2022-06-30	00:50	Regular	Doméstico	Despegue	SAL	AER	AEROLINEAS ARGENTINAS SA	BO-8-737-700	57
28	2022-06-30	00:53	Regular	Doméstico	Aterrizaje	DOZ	AER	JETSMART AIRLINES S.A.	AIB-A320-232	91
29	2022-06-30	00:56	Regular	Doméstico	Aterrizaje	AER	TRH	AEROLINEAS ARGENTINAS SA	BO-737-85H	74
30	2022-06-30	00:56	Regular	Doméstico	Aterrizaje	DOZ	AER	FB LINEAS AEREAS - FLYBOND	BO-737-800	0
31	2022-06-30	00:58	Vuelo Privado con Matricula Nacional	Doméstico	Aterrizaje	FDO	AIA	0	CE-501	1
32	2022-06-30	00:59	Regular	Doméstico	Aterrizaje	EZE	SAL	AEROLINEAS ARGENTINAS SA	EMB-ER190100GW	42
33	2022-06-30	01:00	Regular	Doméstico	Despegue	BAR	EZE	JETSMART AIRLINES S.A.	AIB-A320-232	0
34	2022-06-30	01:01	Regular	Doméstico	Despegue	OSA	TRC	AEROLINEAS ARGENTINAS SA	EMB-ER190100GW	0
35	2022-06-30	01:02	No Regular	Doméstico	Despegue	SHV	AER	BARRES FLY SA	LI-60	3

Se observa que se trajeron 483122 registros (cada uno correspondiente a un tipo de movimiento en particular: Despegue o Aterrizaje) . Estos corresponden solo a los vuelos domésticos.

La lógica de la tabla parece ser la siguiente: si el campo **tipo_de_movimiento** es *Despegue*, el campo **aeropuerto** es el de origen, y el campo **origen-destino** es el destino. Por otro lado, cuando el campo **tipo_de_movimiento** es *Aterrizaje*, el campo **aeropuerto** es el de destino y el campo **origen-destino** es el origen.

Cada registro debería ser interpretado como el correspondiente a un solo tramo de un vuelo. Puede ocurrir que el vuelo total tenga varios tramos (escalas).

Esto debería tenerse en cuenta al momento de filtrar los datos. Para que los campos **tipo_de_movimiento** y **aeropuerto** no dependan del **tipo_de_movimiento** deberemos filtrar en las consultas posteriores como **tipo_de_movimiento = Despegue** para no duplicar vuelos. La elección corresponde a que puede haber vuelos que despegaron en ese intervalo de tiempo y luego aterrizaron al día siguiente (fuera del intervalo).

Tabla aeropuerto_detalle_tabla

	aeropuerto	oac	iata	tipo	denominacion	coordenadas	latitud	longitud	elev	uom	ref	distancia_ref	direccion_ref
1	ACH	[NULL]	[NULL]	Aeródromo	CORONEL BOGADO/AGROSERV	"33°16'20"S 60°34'14"W"	-60.57066	-33.27226	44	Metros	Coronel Bogado	6	NE
2	ACH	[NULL]	[NULL]	Aeródromo	GENERAL ACHA	"37°24'6"S 64°36'49"W"	-64.61351	-37.40164	277	Metros	General Acha	3	SO
3	ACM	[NULL]	[NULL]	Aeródromo	ARRECIFES/LA CURA MALAL	"34°43'33"S 60°8'30"W"	-60.1417	-34.07574	37	Metros	Arrecifes	4	OSO
4	ADO	SAWD	PUD	Aeródromo	PUERTO DESEADO	"47°44'6"S 65°54'15"W"	-65.9041	-47.73511	82	Metros	Puerto Deseado	2	N
5	ADT	[NULL]	[NULL]	Aeródromo	BANDERA/AGROSERVICIOS DOI	"28°51'19"S 62°15'53"W"	-62.26462	-28.85541	75	Metros	Bandera	4	N
6	ADU	[NULL]	[NULL]	Aeródromo	BANDERA/DUTTO	"28°52'1"S 62°14'17"W"	-62.23812	-28.86691	87	Metros	Bandera	3	NE
7	AER	SABE	ASP	Aeródromo	BUENOS AIRES/AEROPARQUE J.	"34°33'32"S 58°24'59"W"	-58.41638889	-34.55888889	5.6	Metros	Ciudad de Buen	2	NE
8	AGI	SAVA	[NULL]	Aeródromo	PIEDRA DEL ÁGUILA	"40°11'32"S 70°03'39"W"	-70.01075	-40.19232	649	Metros	Piedra del Águila	17	SSE
9	AGR	[NULL]	[NULL]	Aeródromo	ALTA GRACIA	"31°39'4"S 64°23'38"W"	-64.39388889	-31.65111111	533	Metros	Alta Gracia	2	E
10	AII	[NULL]	[NULL]	Aeródromo	CHACABUCO/LAS DOS A	"34°48'40"S 60°30'60"W"	-60.51661	-34.81116	60	Metros	Chacabuco	20	SSO
11	AIA	[NULL]	[NULL]	Aeródromo	LA LAJA	"31°21'5"S 68°28'23"W"	-68.47305556	-31.35138889	650	Metros	La Laja	1	SSE
12	ALC	[NULL]	[NULL]	Aeródromo	AMEGHINO/LA CHACRA	"34°48'34"S 62°31'23"W"	-62.52293	-34.80942	109	Metros	Florentino Ame	7	NO
13	ALL	[NULL]	[NULL]	Aeródromo	ALLEN	"38°57'30"S 67°48'10"W"	-67.80277778	-38.95833333	290	Metros	Allen	3.5	E
14	ALR	[NULL]	[NULL]	Aeródromo	ALEJANDRO ROCA	"33°20'55"S 63°43'48"W"	-63.73	-33.34861111	210	Metros	Alejandro Roca	1.5	SO
15	ALT	[NULL]	[NULL]	Aeródromo	CRUZ ALTA	"33°02'8"S 61°50'13"W"	-61.83694444	-33.00777778	97	Metros	Cruz Alta	2	O
16	ALV	[NULL]	[NULL]	Aeródromo	ALVEAR/AGROALVEAR SRL	"33°47'7"S 60°40'11"W"	-60.66682	-33.06853	32	Metros	Alvear	4.5	O
17	AMA	[NULL]	[NULL]	Aeródromo	ROBERTS/LA AMALIA	"35°6'2"S 62°0'60"W"	-62.01657	-35.10043	100	Metros	Roberts	6.5	NO
18	AME	[NULL]	[NULL]	Aeródromo	AMEGHINO/SIGFRIDO ROHR	"34°49'50"S 62°28'41"W"	-62.46787	-34.83057	110	Metros	Florentino Ame	2	N
19	AMG	[NULL]	[NULL]	Aeródromo	AMEGHINO	"34°50'43"S 62°28'58"W"	-62.48266	-34.84541	110	Metros	Florentino Ame	2.5	ONO
20	ANA	[NULL]	[NULL]	Aeródromo	PARANÁ/AEROCUB	"31°45'28"S 60°22'26"W"	-60.37388889	-31.75777778	60	Metros	La Picada	15	ESE
21	AND	[NULL]	[NULL]	Aeródromo	ANDALGALÁ	"27°37'41"S 66°20'34"W"	-66.34277778	-27.62805556	900	Metros	Andalgala	1	O
22	APO	[NULL]	[NULL]	Aeródromo	APÓSTOLES	"27°54'12"S 55°45'56"W"	-55.76555556	-27.90333333	117	Metros	Apóstoles	1.6	NO
23	ARB	[NULL]	[NULL]	Aeródromo	ARRIBEÑOS	"34°13'19"S 61°23'12"W"	-61.38666667	-34.22194444	82	Metros	Arribeños	3.3	SO
24	ARS	SAVR	ARR	Aeródromo	ALTO RÍO SENGUE/R/D. C. SZLA	"45°04'8"S 70°48'46"W"	-70.81282	-45.0132	697	Metros	Alto Río Sengue	3	N
25	ARY	[NULL]	[NULL]	Aeródromo	ARROYITO/ARCOR	"31°25'20"S 63°01'8"W"	-63.0005	-31.42222222	148	Metros	Arroyito	4	E
26	ATE	[NULL]	[NULL]	Aeródromo	ZARATE	"34°7'6"S 59°45'8"W"	-59.08266	-34.11829	26	Metros	Zarate	6	SO
27	ATN	[NULL]	[NULL]	Aeródromo	TRENQUE LAUQUEN/LA ARGEN	"35°59'52"S 62°42'23"W"	-62.70635	-35.99773	95	Metros	Trenque Lauque	2	SE
28	AUL	[NULL]	[NULL]	Aeródromo	AMEGHINO/AUGUSTO LEGUIZA	"34°49'5"S 62°28'38"W"	-62.47709	-34.81797	109	Metros	Florentino Ame	3	NNO
29	AVA	[NULL]	[NULL]	Aeródromo	ALVEAR/AEROPARQUE ROSARI	"33°24'9"S 60°35'51"W"	-60.59756	-33.04691	26	Metros	Alvear	2.5	ENE
30	AVL	[NULL]	[NULL]	Aeródromo	AVELLANEDA/PRESIDENTE AVEI	"29°6'22"S 59°39'27"W"	-59.6575	-29.10611111	45	Metros	Avellaneda	1	N

Se observa que se trajeron 693 registros. Estos corresponden a todos los aeropuertos del país.

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

The screenshot shows the DBeaver 24.1.4 interface. The left sidebar displays a database navigator with 'localhost - localhost:3306' selected. The main editor shows a SQL query in a script editor:

```
-- Query Ejercicio 1 - ej 6 - Trabajo final

SELECT COUNT(*) AS total_vuelos
FROM aeropuertos.aeropuerto_tabla
WHERE (fecha BETWEEN '2021-12-01' AND '2022-01-31') and (tipo_de_movimiento == 'Despegue')
```

Below the query, the 'Resultados 1' tab shows the results in a grid view:

	total_vuelos
1	29.138

La cantidad total de vuelos domésticos que despegaron en ese intervalo de tiempo es **29138**.

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

The screenshot shows the DBeaver 24.1.4 interface. The left sidebar displays a database navigator with 'localhost - localhost:3306' selected. The main editor shows a SQL query in a script editor:

```
-- Query Ejercicio 1 - ej 7 - Trabajo final

SELECT COUNT(pasajeros) AS total_pasajeros
FROM aeropuertos.aeropuerto_tabla
WHERE (fecha BETWEEN '2021-01-01' AND '2022-06-30') and (aerolinea_nombre == 'AEROLINEAS ARGENTINAS SA') and (tipo_de_movimiento == 'Despegue')
```

Below the query, the 'Resultados 1' tab shows the results in a grid view:

	total_pasajeros
1	74.331

La cantidad total de pasajeros que viajaron en el lapso estipulado por Aerolíneas Argentinas fue **74331**.

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

DBEaver 24.1.4 - <localhost 2> Script-27

Archivo Editar Navegar Buscar Editor SQL Base de Datos Ventana Ayuda

Navegador de Bases de Datos

Ingrese parte del nombre de un objeto aquí

- DBEaver Sample Database (SQLite)
 - localhost - localhost:3306
 - localhost 2 - localhost:10000
 - aeropuertos
 - default
 - f1
 - northwind_analytics
 - titanic
 - tripdata
 - trips
 - northwind - localhost:5432

```
-- Query Ejercicio 1 - ej 8 - Trabajo final

SELECT
  at2.fecha,
  at2.horautc AS hora,
  at2.aeropuerto AS codigo_aeropuerto_salida,
  at2_salida.ref AS ciudad_de_salida,
  at2.origen_destino AS codigo_aeropuerto_destino,
  at2_arribo.ref AS ciudad_de_arribo,
  at2.pasajeros AS cantidad_pasajeros_por_vuelo
FROM
  aeropuertos.aeropuerto_tabla at2
LEFT JOIN aeropuertos.aeropuerto_detalle at2_salida
ON at2.aeropuerto = at2_salida.aeropuerto -- datos del aeropuerto de salida
LEFT JOIN aeropuertos.aeropuerto_detalle at2_arribo
ON at2.origen_destino = at2_arribo.aeropuerto -- datos del aeropuerto de arribo
WHERE (at2.fecha BETWEEN '2022-01-01' AND '2022-06-30') AND (tipo_de_movimiento == 'Despegue')
ORDER BY at2.fecha DESC
```

Resultados 1

SELECT at2.fecha, at2.horautc AS hora, at2.aeropuerto

	fecha	hora	codigo_aeropuerto_salida	ciudad_de_salida	codigo_aeropuerto_destino	ciudad_de_arribo	cantidad_pasajeros_por_vuelo
1	2022-06-30	19:28	BCA	Bahía Blanca	FDO	San Fernando	0
2	2022-06-30	19:28	USU	Ushuaia	AER	Ciudad de Buenos Aires	0
3	2022-06-30	19:29	CRR	Corrientes	SIS	Resistencia	0
4	2022-06-30	19:30	CBA	Córdoba	SRC	Santa Rosa del Conlara	0
5	2022-06-30	19:32	AER	Ciudad de Buenos Aires	USU	Ushuaia	84
6	2022-06-30	19:32	FDO	San Fernando	BOS	Lobos	0
7	2022-06-30	19:32	ROS	Rosario	IGU	Cataratas del Iguazú	0
8	2022-06-30	19:36	CBA	Córdoba	DOZ	Mendoza	47
9	2022-06-30	19:36	FDO	San Fernando	FDO	San Fernando	0
10	2022-06-30	19:37	AER	Ciudad de Buenos Aires	CRR	Corrientes	2
11	2022-06-30	19:38	CRR	Corrientes	CRR	Corrientes	0
12	2022-06-30	19:39	CRV	Comodoro Rivadavia	NEU	Neuquén	0
13	2022-06-30	19:40	PAL	El Palomar	PAL	El Palomar	0

200 row(s) fetched - 9s (0,020s fetch), on 2025-11-16 at 22:04:26

El total de registros ordenados en forma descendente en el lapso considerado es **95877**.

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

DBEaver 24.1.4 - <localhost 2> Script-27

Archivo Editar Navegar Buscar Editor SQL Base de Datos Ventana Ayuda

Navegador de Bases de Datos

Ingrese parte del nombre de un objeto aquí

- DBEaver Sample Database (SQLite)
 - localhost - localhost:3306
 - localhost 2 - localhost:10000
 - aeropuertos
 - car_rental_db
 - default
 - f1
 - northwind_analytics
 - titanic
 - tripdata
 - trips
 - northwind - localhost:5432

```
select aerolinea_nombre as nombre_aerolinea, sum(pasajeros) as cantidad_pasajeros
FROM aeropuertos.aeropuerto_tabla at2
WHERE (fecha BETWEEN '2021-01-01' and '2022-06-30') AND (aerolinea_nombre is not null) AND (aerolinea_nombre <> '')
group by aerolinea_nombre
order by cantidad_pasajeros DESC
limit 10
```

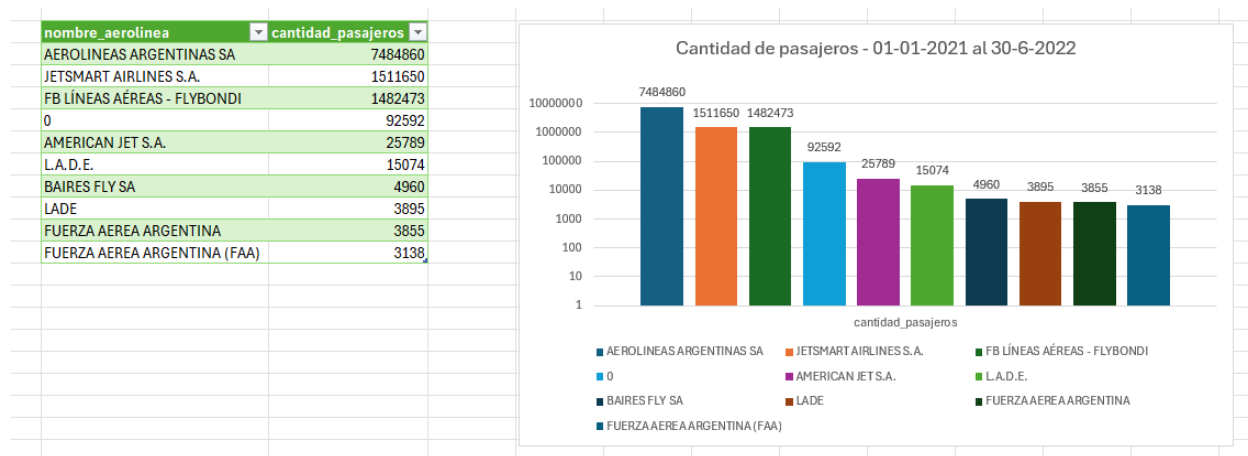
Resultados 1

select aerolinea_nombre as nombre_aerolinea

	nombre_aerolinea	cantidad_pasajeros
1	AEROLINEAS ARGENTINAS SA	7.484.860
2	JETSMART AIRLINES S.A.	1.511.650
3	FB LINEAS AEREAS - FLYBONDI	1.482.473
4	0	92.592
5	AMERICAN JET S.A.	25.789
6	L.A.D.E.	15.074
7	BAIRES FLY SA	4.960
8	LADE	3.895
9	FUERZA AEREA ARGENTINA	3.855
10	FUERZA AEREA ARGENTINA (FAA)	3.138

Para la visualización, se procedió a exportar la tabla generada por la consulta en DBeaver (Tabla ejercicio 1 ej9 - Trabajo final.csv) para luego importarla en Excel como csv.

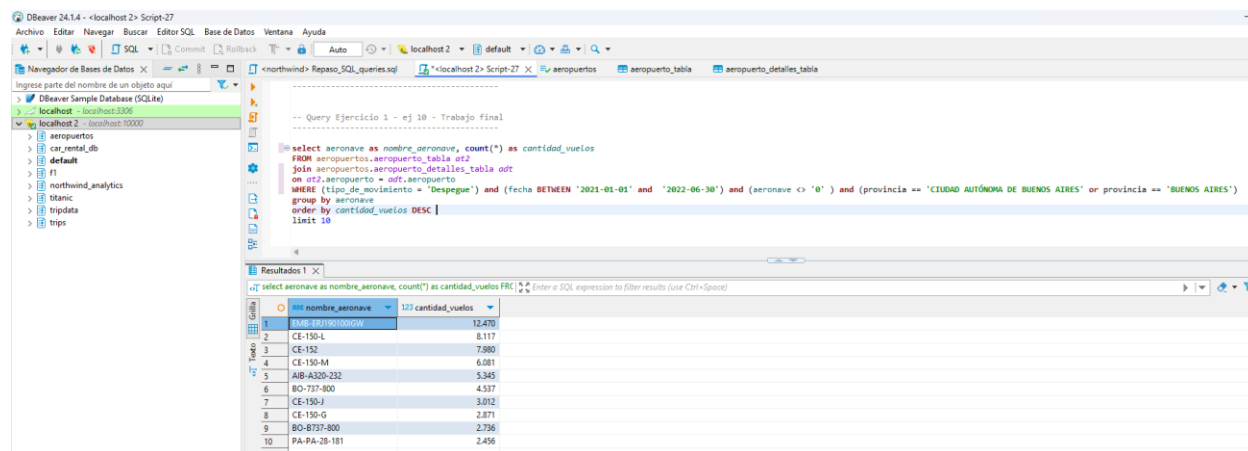
Una vez allí, se procedió a insertar gráfico y personalizarlo:



Aerolíneas Argentinas concentra cerca del 70 % del mercado de transporte aéreo de pasajeros y, en conjunto con JetSmart y Flybondi, alcanza aproximadamente el 98 % de dicho mercado.

En la visualización se utilizó escala logarítmica, dada la discrepancia en los órdenes de magnitud dentro de la variable considerada.

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



Exportamos (igual que recién) la tabla generada por la consulta en DBeaver (**Tabla ejercicio 1 ej 10 - Trabajo final.csv**) para luego importarla en Excel como csv.



Se observa que el modelo de aeronave predominante en la cantidad de despegues de BA y CABA en el lapso considerado es **EMB – ERJ190100IGW**, con 12470 despegues.

11. Qué datos externos agregaría en este dataset que mejoraría el análisis de los datos

Considero que existen varios aspectos para mejorar el dataset:

- En relación con la documentación de los campos, no resulta trivial determinar el significado de algunos de ellos. Por ejemplo, más arriba incorporé una discusión específica sobre la interpretación de ciertos campos, con el fin de evitar análisis erróneos derivados de la falta de documentación adecuada.
- En términos campos a agregar, por ejemplo, **código de vuelo**, para seguimiento; **aeropuerto_origen_codigo** y **aeropuerto_destino_codigo** separados, para facilitar el análisis y campos asociados a **fechas de salidas y llegadas programadas y reales**, para analizar demoras y congestiones.

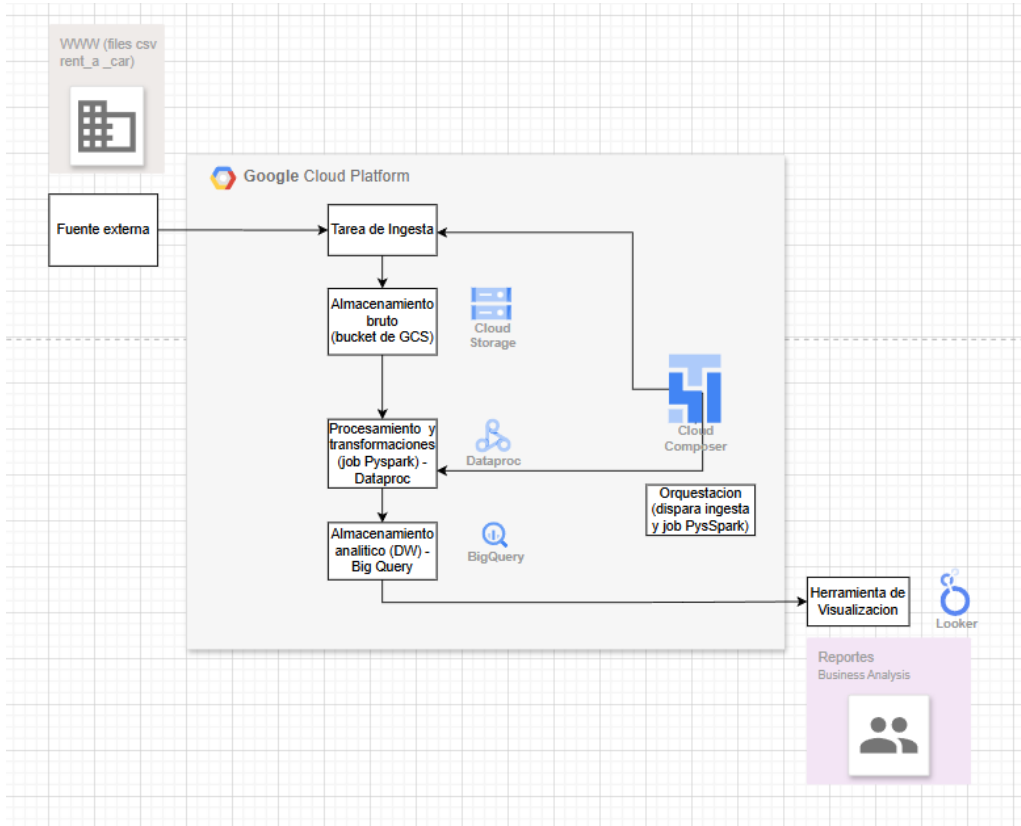
12. Elabore sus conclusiones y recomendaciones sobre este proyecto.

Para elaborar informes más completos, resulta necesario, en primer lugar, precisar con mayor claridad las necesidades del Ministerio de Transporte y, a partir de ello, solicitar —en la medida de lo posible— acceso a información complementaria (con la que ANAC probablemente ya cuenta) sobre los vuelos en forma desagregada. Esta información permitiría enriquecer la base de datos publicada, incorporando las sugerencias indicadas en el punto 11) y otras dimensiones de análisis que pudieran surgir de los requerimientos planteados, las cuales se traducirían en nuevas variables a relevar. La construcción de KPI's (indicadores) específicos de la actividad a partir de las mismas puede generar información de valor.

Recomendaría también la implementación de una infraestructura de datos que permita un seguimiento en tiempo real de KPI's críticos del sistema que permita mejorar la toma de decisiones. De todas maneras, para datos de uso público no desestimaría la publicación de datasets mensuales (aunque mejorados).

13. Proponer una arquitectura alternativa para este proceso ya sea con herramientas on premise o cloud (Sí aplica)

Proponemos la siguiente arquitectura, basada en Google Cloud Platform (GCP):



La tarea de ingesta se hace a través de un archivo similar a **ingest_ejercicio_1.sh** alojado en un bucket de Google Cloud Storage. Este proceso lo dispara Cloud Composer.

El job de PySpark (archivo similar a **transform_load_7.py**) lo ejecuta Dataproc y también es disparado por Cloud Composer.

Las tareas de consulta en SQL se realizan dentro de Big Query y finalmente, desde afuera de GCP, podemos conectarnos con alguna herramienta de visualización como Looker, para generar reportes (Dashboards).