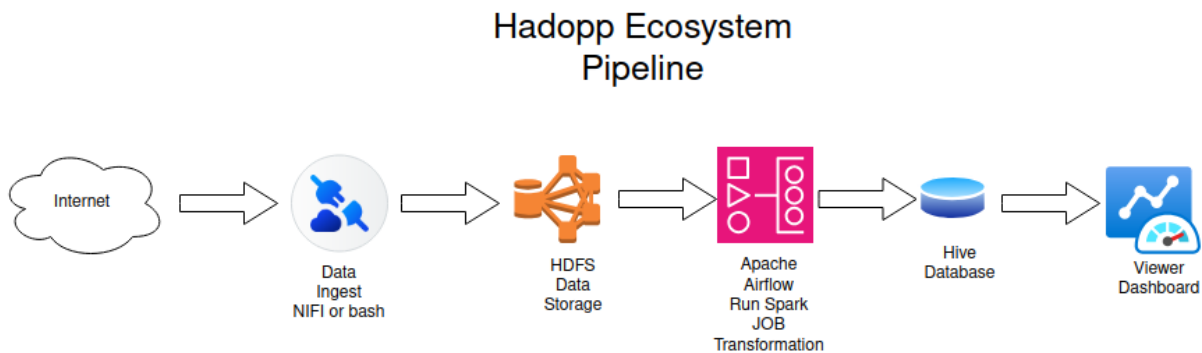


ESCUADERO
BOOTCAM DATA ENGINEERING
EXAMEN FINAL
REV 12-01-25

EJ. 1 AVIACIÓN CIVIL

En primera instancia realizaré el análisis a través de las herramientas del ecosistema Hadoop según el siguiente pipeline:



1) INGESTA DE DATA

CREAMOS UN DIRECTORIO EN HDFS PARA INGESTAR LOS ARCHIVOS DE ANÁLISIS

`hdfs dfs -mkdir -p ingest3/`

```
hadoop@d4236fd64627:~$ hdfs dfs -mkdir -p ingest/
hadoop@d4236fd64627:~$ hdfs dfs -ls
Found 3 items
drwxr-xr-x  - hadoop supergroup          0 2024-12-16 18:47 sparkStaging
drwxr-xr-x  - hadoop supergroup          0 2024-12-18 14:36 ingest
drwxr-xr-x  - hadoop supergroup          0 2022-01-22 23:56 inputs
hadoop@d4236fd64627:~$
```

Ingestamos los csv a HDFS

`wget -P /home/hadoop/landing 2021-informe-ministerio.csv`

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

`wget -P /home/hadoop/landing 202206-informe-ministerio.csv`

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

`wget -P /home/hadoop/landing aeropuertos_detalle.csv`

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

ingesta completada en HDFS

```

hadoop@4236fd64627:~$ wget -P 202206-informe-ministerio.csv "https://dataengineerpublic.blob.core.windows.net/data-engineer/202206-informe-ministerio.csv"
--2024-12-18 14:47:34-- https://dataengineerpublic.blob.core.windows.net/data-engineer/202206-informe-ministerio.csv
Resolving dataengineerpublic.blob.core.windows.net (dataengineerpublic.blob.core.windows.net)... 20.150.25.164
Connecting to dataengineerpublic.blob.core.windows.net (dataengineerpublic.blob.core.windows.net)|20.150.25.164|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 22833520 (22M) [text/csv]
Saving to: '202206-informe-ministerio.csv/202206-informe-ministerio.csv'

202206-informe-ministerio.csv      100%[=====] 21.78M  495KB/s   in 37s

2024-12-18 14:48:12 (602 KB/s) - '202206-informe-ministerio.csv/202206-informe-ministerio.csv' saved [22833520/22833520]

hadoop@4236fd64627:~$ wget -P aeropuertos_detalle.csv "https://dataengineerpublic.blob.core.windows.net/data-engineer/aeropuertos_detalle.csv"
--2024-12-18 14:49:03-- https://dataengineerpublic.blob.core.windows.net/data-engineer/aeropuertos_detalle.csv
Resolving dataengineerpublic.blob.core.windows.net (dataengineerpublic.blob.core.windows.net)... 20.150.25.164
Connecting to dataengineerpublic.blob.core.windows.net (dataengineerpublic.blob.core.windows.net)|20.150.25.164|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 136007 (133K) [text/csv]
Saving to: 'aeropuertos_detalle.csv/aeropuertos_detalle.csv'

aeropuertos_detalle.csv          100%[=====] 132.82K  213KB/s   in 0.6s

2024-12-18 14:49:04 (213 KB/s) - 'aeropuertos_detalle.csv/aeropuertos_detalle.csv' saved [136007/136007]

hadoop@4236fd64627:~$

```

Subimos los archivos a HDFS

```

hdfs dfs -put 2021-informe-ministerio.csv /ingest/
hdfs dfs -put 202206-informe-ministerio.csv /ingest/
hdfs dfs -put aeropuertos_detalle.csv /ingest/

```

desde el directorio landing

```

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/

```

Verificamos si los datos están

```

hdfs dfs -ls /ingest/

```

```

hadoop@4236fd64627:~$ hdfs dfs -put 2021-informe-ministerio.csv /ingest/
hadoop@4236fd64627:~$ hdfs dfs -put 202206-informe-ministerio.csv /ingest/
hadoop@4236fd64627:~$ hdfs dfs -put aeropuertos_detalle.csv /ingest/
hadoop@4236fd64627:~$ hdfs dfs -ls /ingest/
Found 4 items
drwxr-xr-x - hadoop hadoop      0 2024-12-18 15:00 /ingest/2021-informe-ministerio.csv
drwxr-xr-x - hadoop hadoop      0 2024-12-18 15:00 /ingest/202206-informe-ministerio.csv
drwxr-xr-x - hadoop hadoop      0 2024-12-18 15:00 /ingest/aeropuertos_detalle.csv
-rwxrwx-- 3 hadoop hadoop    30 2024-12-12 12:48 /ingest/part-00000-c182575e-175a-4a0b-8170-9fa9cb98deee-c000
hadoop@4236fd64627:~$

```

--

SCRIPT de ingesta:

```

#Escudero final ejercicio 1 -Aviación Civil-
#!/bin/bash

```

```

# Mensaje de inicio
echo "***** Inicio Ingesta Aviacion Civil *****"

```

```

# 1 Crear el directorio 'landing' en /home/hadoop
mkdir -p /home/hadoop/landing

```

```

# 2 Eliminar todos los archivos en el directorio local 'landing'
rm -f /home/hadoop/landing/*

```

3 Descargar los archivos desde las URLs especificadas

```
wget -P /home/hadoop/landing "https://dataengineerpublic.blob.core.windows.net/data-engineer/2021-informe-ministerio.csv"
```

```
wget -P /home/hadoop/landing "https://dataengineerpublic.blob.core.windows.net/data-engineer/202206-informe-ministerio.csv"
```

```
wget -P /home/hadoop/landing "https://dataengineerpublic.blob.core.windows.net/data-engineer/aeropuertos_detalle.csv"
```

4 Eliminar todos los archivos en el directorio HDFS '/ingest'

```
hdfs dfs -rm -f /ingest/*
```

5 Subir los nuevos archivos desde 'landing' a HDFS '/ingest'

```
hdfs dfs -put /home/hadoop/landing/* /ingest/
```

Mensaje de finalizacion

```
echo "\n***** Fin Ingesta Aviacion Civil *****"
```

Creamos el script por consola en un .sh

nano /home/hadoop/scripts/data-aviacion.sh

y debemos darle permiso de ejecución

chmod +x /home/hadoop/scripts/data-aviacion.sh

lo ejecutamos para verificar el funcionamiento y chequeemos la carga de los datos

bash /home/hadoop/scripts/data-aviacion.sh

el bash está funcionando

```
hadoop@4236fd64627:~$ bash /home/hadoop/scripts/data-aviacion.sh
--2024-12-18 17:31:00-- https://dataengineerpublic.blob.core.windows.net/data-engineer/2021-informe-ministerio.csv
Resolving dataengineerpublic.blob.core.windows.net (dataengineerpublic.blob.core.windows.net)... 20.150.25.164
Connecting to dataengineerpublic.blob.core.windows.net (dataengineerpublic.blob.core.windows.net)|20.150.25.164|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 32322556 (31M) [text/csv]
Saving to: '/home/hadoop/landing/2021-informe-ministerio.csv'

/home/hadoop/landing/2021-informe-ministerio.csv 100%[=====] 30.82M 668KB/s ln 53s

2024-12-18 17:31:54 (599 KB/s) - '/home/hadoop/landing/2021-informe-ministerio.csv' saved [32322556/32322556]

--2024-12-18 17:31:54-- https://dataengineerpublic.blob.core.windows.net/data-engineer/202206-informe-ministerio.csv
Resolving dataengineerpublic.blob.core.windows.net (dataengineerpublic.blob.core.windows.net)... 20.150.25.164
Connecting to dataengineerpublic.blob.core.windows.net (dataengineerpublic.blob.core.windows.net)|20.150.25.164|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 22833520 (22M) [text/csv]
Saving to: '/home/hadoop/landing/202206-informe-ministerio.csv'

/home/hadoop/landing/202206-informe-ministerio.csv 100%[=====] 21.78M 657KB/s ln 35s

2024-12-18 17:32:30 (632 KB/s) - '/home/hadoop/landing/202206-informe-ministerio.csv' saved [22833520/22833520]

--2024-12-18 17:32:30-- https://dataengineerpublic.blob.core.windows.net/data-engineer/aeropuertos_detalle.csv
Resolving dataengineerpublic.blob.core.windows.net (dataengineerpublic.blob.core.windows.net)... 20.150.25.164
Connecting to dataengineerpublic.blob.core.windows.net (dataengineerpublic.blob.core.windows.net)|20.150.25.164|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 136007 (133K) [text/csv]
Saving to: '/home/hadoop/landing/aeropuertos_detalle.csv'

/home/hadoop/landing/aeropuertos_detalle.csv 100%[=====] 132.82K 247KB/s ln 0.5s

2024-12-18 17:32:32 (247 KB/s) - '/home/hadoop/landing/aeropuertos_detalle.csv' saved [136007/136007]

rm: '/ingest/2021-informe-ministerio.csv': Is a directory
rm: '/ingest/202206-informe-ministerio.csv': Is a directory
rm: '/ingest/aeropuertos_detalle.csv': Is a directory
Deleted '/ingest/part-00000-c102575e-175a-4a0b-8170-9fa9cb98deee-c000'
put: '/ingest/2021-informe-ministerio.csv': File exists
put: '/ingest/202206-informe-ministerio.csv': File exists
put: '/ingest/aeropuertos_detalle.csv': File exists
```

--
y se cargaron los archivos en HDFS

```
hadoop@d4236fd64627:~$ hdfs dfs -ls /ingest
Found 3 items
drwxr-xr-x - hadoop hadoop      0 2024-12-18 15:00 /ingest/2021-informe-ministerio.csv
drwxr-xr-x - hadoop hadoop      0 2024-12-18 15:00 /ingest/202206-informe-ministerio.csv
drwxr-xr-x - hadoop hadoop      0 2024-12-18 15:00 /ingest/aeropuertos_detalle.csv
hadoop@d4236fd64627:~$
```

Damos los permisos necesarios de los archivos

`hdfs dfs -chmod 644 /ingest/.csv`*

y del directorios

`hdfs dfs -chmod 755 /ingest`

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

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

(aeropuertos_detalle.csv)

pasos previos

```
show databases;  
con esto vemos que tenemos
```

creamos una nueva data base

```
-- create database vuelosdb;
```

le indico en que DB voy a trabajar

```
-- use vuelosdb;
```

para saber donde estamos parados

```
-- select current_database();
```

Creación de tablas:

1. En Hive, crear la siguiente tabla (externa) en la base de datos vuelos data:

```
CREATE EXTERNAL TABLE aeropuerto_tabla (  
    fecha DATE,  
    hora_utc 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 ','  
STORED AS TEXTFILE  
LOCATION '/ingest';
```

TABLA creada en Hive

```
hive> describe aeropuerto_tabla;
OK
fecha                date
hora_utc             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
Time taken: 0.041 seconds, Fetched: 10 row(s)
```

a. tabla aeropuerto_detalle_tabla

```
CREATE EXTERNAL TABLE 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
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/ingest';
```

```

hive> describe aeropuerto_detalle;
OK
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
Time taken: 0.058 seconds, Fetched: 17 row(s)

```

Dar permiso de acceso en HDFS para que se puedan leer los archivos en /ingest

```
hdfs dfs -chmod -R 555 /ingest
```

3.- Ingesta de archivos con pyspark y armado del DF total para la transformación

Generamos el dag_vuelos.py para la ejecucion en airflow, código:

```

#ESCUADERO FINAL EJERCICIO 1 DGA
from datetime import timedelta
from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.dummy import DummyOperator
from airflow.utils.dates import days_ago

args = {
    'owner': 'airflow',
}

with DAG(
    dag_id='VUELOS-DAG',

```

```
default_args=args,
schedule_interval='0 0 * * *',
start_date=days_ago(1),
catchup=False,
dagrun_timeout=timedelta(minutes=60),
tags=['ingest', 'transform'],
params={"example_key": "example_value"},
) as dag:
```

```
inicia_proceso = DummyOperator(
    task_id='inicia_proceso',
)
```

```
finaliza_proceso = DummyOperator(
    task_id='finaliza_proceso',
)
```

```
ingest = BashOperator(
    task_id='ingesta',
    bash_command='/usr/bin/sh /home/hadoop/scripts/data-aviacion.sh ',
)
```

```
transform = BashOperator(
    task_id='transformacion',
    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/vuelos_transformacion.py ',
)
```

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

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



```
GNU nano 4.8 dag_vuelos.py
#DAG para el ejercicio 1 de la semana 10
from datetime import timedelta
from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.dummy import DummyOperator
from airflow.utils.dates import days_ago

args = {
    'owner': 'airflow',
}

with DAG(
    dag_id='VUELOS-DAG',
    default_args=args,
    schedule_interval='0 0 * * *',
    start_date=days_ago(1),
    catchup=False,
    dagrun_timeout=timedelta(minutes=60),
    tags=['ingest', 'transform'],
    params={'example_key': "example_value"},
) as dag:
    inicia_proceso = DummyOperator(
        task_id='inicia_proceso',
    )

    finaliza_proceso = DummyOperator(
        task_id='finaliza_proceso',
    )

    ingest = BashOperator(
        task_id='ingesta',
        bash_command='/usr/bin/sh /home/hadoop/scripts/data_aviacion.sh ',
    )

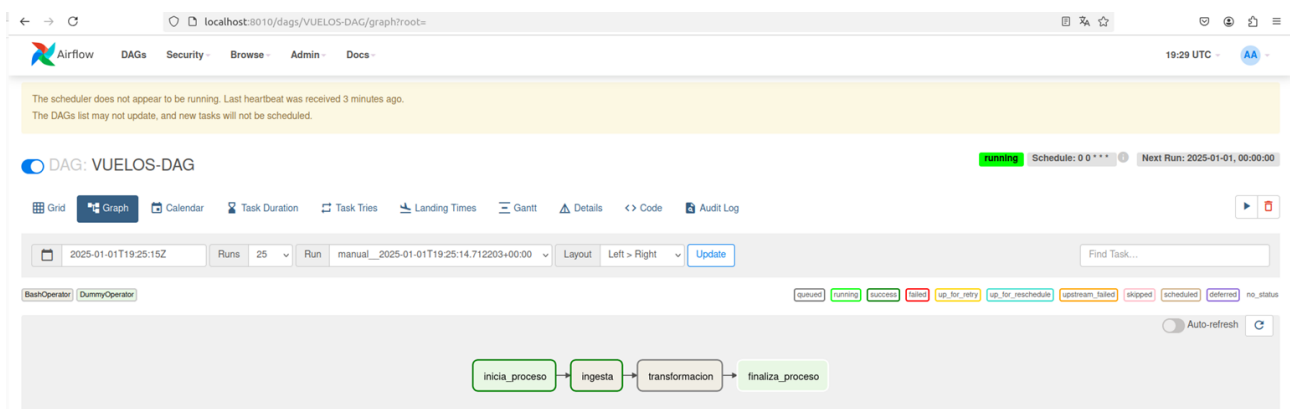
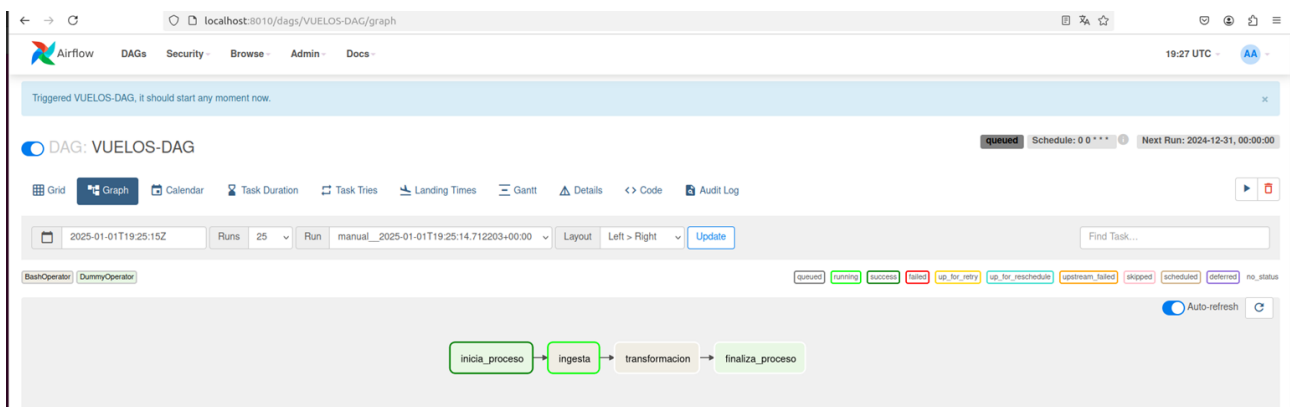
    transform = BashOperator(
        task_id='transformacion',
        bash_command='ssh hadoop@172.17.0.2 /home/hadoop/spark/bin/spark-submit --files /home/hadoop/hive/conf/hive-site.xml /home/hadoop/scripts/vuelos_transformacion.py ',
    )

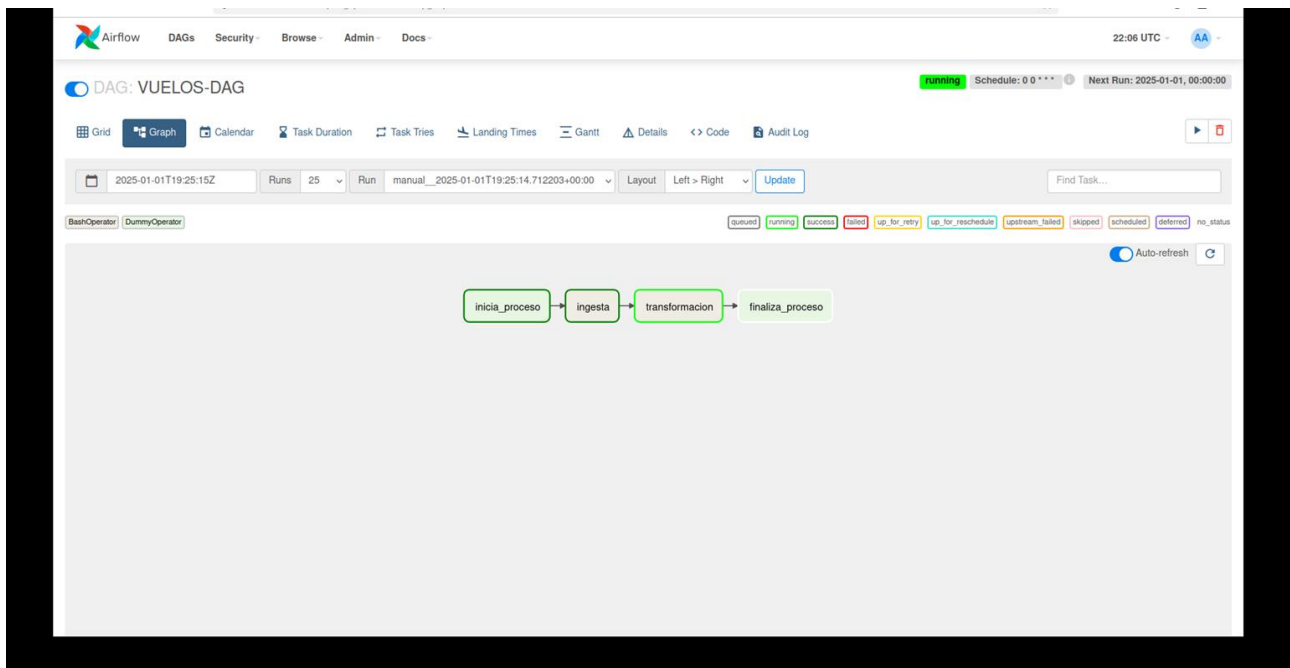
    inicia_proceso >> ingest >> transform >> finaliza_proceso

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

```
hadoop@4236fd64627:~/airflow/dags$ ls
DAG_str_weather.py  DAG_weather2kafka.py  __pycache__  dag_vuelos.py  example-DAG.py  ingest-transform.py  kika-DAG.py  kika-DAG2.py  kika-DAG3.py
hadoop@4236fd64627:~/airflow/dags$
```

en airflow





4.- Realizar las siguiente transformaciones en los pipelines de datos:

```
hadoop@d4236fd64627:~/scripts$ nano vuelos_transformacion.py
hadoop@d4236fd64627:~/scripts$ ls
aeropuerto_transform.py  data_avlacion.sh  derby.log  ingest.sh  spark-warehouse  start-services.sh  transformation.py  vuelos_transformacion.py
hadoop@d4236fd64627:~/scripts$
```

--

Creamos un vuelos_transformacion.py para las transformaciones de los datos

ESCUDERO FINAL ejercicio 1 transformación

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import lit
from pyspark.sql import functions as F
from pyspark.sql.types import FloatType
```

0. Configuración de SparkSession con soporte para Hive

```
spark = SparkSession.builder \
    .appName("Airport Trips Processing") \
    .enableHiveSupport() \
    .getOrCreate()
```

1. Leemos los archivos Parquet y creamos el DataFrame df_airport

```
df_2021 = spark.read.option("header", "true").option("sep", ";").csv("/ingest/2021-informe-ministerio.csv")
df_2022 = spark.read.option("header", "true").option("sep", ";").csv("/ingest/202206-informe-ministerio.csv")
```

```
df_aeroporto = spark.read.option("header", "true").option("sep",
";").csv("/ingest/aeropuertos_detalle.csv")
```

```
# Eliminar columna "calidad_del_dato" de df_2021 y df_2022
df_2021 = df_2021.drop("calidad dato")
df_2022 = df_2022.drop("calidad dato")
```

```
# 2. Union df_2021 y df_2022
```

```
df_vuelos = df_2021.union(df_2022)
```

```
# Eliminar columnas "inhab" y "fir" de df_aeroporto
df_aeroporto = df_aeroporto.drop("inhab", "fir")
```

```
# hay que renombrar algunas col de aeroporto local >> aeropuerto y oaci >>>> aoc
```

```
df_aeroporto = df_aeroporto.withColumnRenamed("local",
"aeropuerto").withColumnRenamed("oaci", "aoc")
```

```
# Filtra por vuelos domesticos, elimina columna 'calidad del dato' y reemplaza nulos
de la columna 'pasajeros' por ceros
```

```
df_vuelos_mod = df_vuelos \
    .withColumn("Pasajeros", F.col("Pasajeros").cast("int")) \
    .withColumn("Fecha", F.to_date(df_vuelos["Fecha"],
"dd/MM/yyyy").alias("Fecha")) \
    .replace({'Doméstico': 'Domestico'}, subset=['Clasificación Vuelo'])
```

```
vuelos_filtered = df_vuelos_mod \
    .filter(df_vuelos_mod['Clasificación Vuelo'] == "Domestico") \
    .fillna(0, 'pasajeros')
```

```
# Renombrar las columnas para que coincidan con las de la tabla de Hive
```

```
vuelos_filtered_renamed = vuelos_filtered \
    .withColumnRenamed("Fecha", "fecha") \
    .withColumnRenamed("Hora UTC", "hora_utc") \
    .withColumnRenamed("Clase de Vuelo (todos los vuelos)", "clase_de_vuelo") \
    .withColumnRenamed("Clasificación Vuelo", "clasificacion_de_vuelo") \
    .withColumnRenamed("Tipo de Movimiento", "tipo_de_movimiento") \
    .withColumnRenamed("Aeropuerto", "aeropuerto") \
    .withColumnRenamed("Origen / Destino", "origen_destino") \
    .withColumnRenamed("Aerolinea Nombre", "aerolinea_nombre") \
    .withColumnRenamed("Aeronave", "aeronave") \
```

```

.withColumnRenamed("Pasajeros", "pasajeros")

# Inserta tabla "vuelos_filtered" en la BD vuelosdb, en la tabla 'aeropuerto_tabla'
funcionando 30-12
vuelos_filtered_renamed.write.insertInto("vuelosdb.aeropuerto_tabla",
overwrite=False)

# reemplaza nulos de la columna 'distancia_ref' por ceros.
df_aeroporto = df_aeroporto\
    .fillna(0, "distancia_ref")

# Convertir elev y distancia_ref a float
df_aeroporto = df_aeroporto.withColumn("elev", df_aeroporto["elev"].cast(FloatType()))
df_aeroporto = df_aeroporto.withColumn("distancia_ref",
df_aeroporto["distancia_ref"].cast(FloatType()))

# reorganizacion de columnas

df_aeroporto_adjusted = df_aeroporto.withColumnRenamed("aoc", "oac")

# Seleccionar únicamente las columnas necesarias para la tabla Hive
columns_to_select = [
    "aeropuerto", "oac", "iata", "tipo", "denominacion", "coordenadas",
    "latitud", "longitud", "elev", "uom_elev", "ref", "distancia_ref",
    "direccion_ref", "condicion", "control", "region", "uso"
]
df_aeroporto_filtered = df_aeroporto_adjusted.select(columns_to_select)

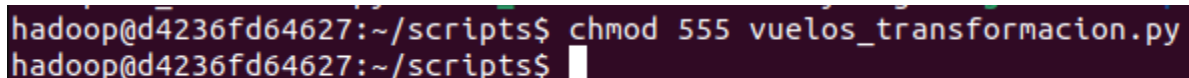
#Insertamos df_aeroporto en la DB vuelosdb tabla aeropurtos

df_aeroporto_filtered.write.insertInto("vuelosdb.aeropuerto_detalle_tabla")

```

Y le damos los permisos de ejecución:

```
chmod 555 vuelos_transformacion.py
```



```

hadoop@d4236fd64627:~/scripts$ chmod 555 vuelos_transformacion.py
hadoop@d4236fd64627:~/scripts$

```

Ejecutar por consola transform

```
/home/hadoop/spark/bin/spark-submit \  
--files /home/hadoop/hive/conf/hive-site.xml \  
/home/hadoop/scripts/vuelos_transformacion.py
```

```
df_aeroporto_adjusted.write.insertInto("vuelosdb.aeroporto_detalle_tabla")
```

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.)

Tabla aeropuerto_detalle_tabla

```
hive> describe aeropuerto_detalle_tabla;  
OK  
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  
Time taken: 0.058 seconds, Fetched: 17 row(s)
```

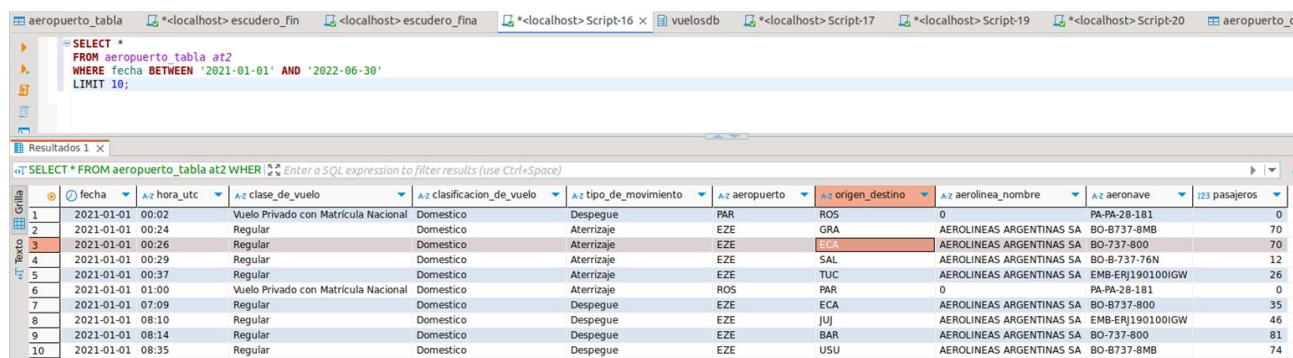
tabla aeropuerto_detalle_tabla

```
Time taken: 1.103 seconds, Fetched: 10 row(s)
hive> describe aeropuerto_tabla;
OK
fecha                                date
hora_utc                            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
Time taken: 0.044 seconds, Fetched: 10 row(s)
hive>
```

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

Desde Postgres hacemos las querries, para contar el numero de vuelos cuento las horas registradas que corresponden a un despegue o aterrizaje

Detalle de aeropuerto tabla



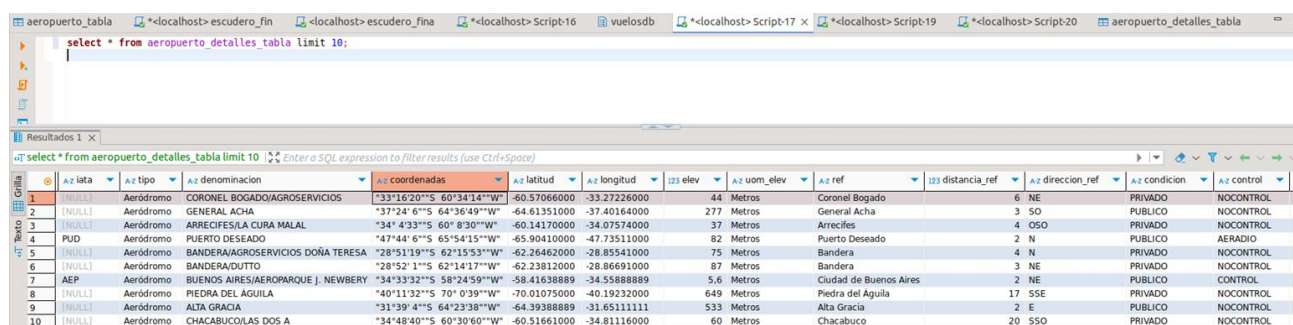
The screenshot shows a PostgreSQL query window with the following SQL query:

```
SELECT *
FROM aeropuerto_tabla at2
WHERE fecha BETWEEN '2021-01-01' AND '2022-06-30'
LIMIT 10;
```

The results are displayed in a table with 11 columns: fecha, hora_utc, clase_de_vuelo, clasificacion_de_vuelo, tipo_de_movimiento, aeropuerto, origen_destino, aerolinea_nombre, aeronave, and pasajeros. The table contains 10 rows of data.

	fecha	hora_utc	clase_de_vuelo	clasificacion_de_vuelo	tipo_de_movimiento	aeropuerto	origen_destino	aerolinea_nombre	aeronave	pasajeros
1	2021-01-01	00:02	Vuelo Privado con Matrícula Nacional	Domestico	Despegue	PAR	ROS	0	PA-PA-28-181	0
2	2021-01-01	00:24	Regular	Domestico	Aterrizaje	EZE	GRA	AEROLINEAS ARGENTINAS SA	BO-8737-8MB	70
3	2021-01-01	00:26	Regular	Domestico	Aterrizaje	EZE	ECA	AEROLINEAS ARGENTINAS SA	BO-737-800	70
4	2021-01-01	00:29	Regular	Domestico	Aterrizaje	EZE	SAL	AEROLINEAS ARGENTINAS SA	BO-8-737-76N	12
5	2021-01-01	00:37	Regular	Domestico	Aterrizaje	EZE	TUC	AEROLINEAS ARGENTINAS SA	EMB-ERJ190100IGW	26
6	2021-01-01	01:00	Vuelo Privado con Matrícula Nacional	Domestico	Aterrizaje	ROS	PAR	0	PA-PA-28-181	0
7	2021-01-01	07:09	Regular	Domestico	Despegue	EZE	ECA	AEROLINEAS ARGENTINAS SA	BO-8737-800	35
8	2021-01-01	08:10	Regular	Domestico	Despegue	EZE	JUJ	AEROLINEAS ARGENTINAS SA	EMB-ERJ190100IGW	46
9	2021-01-01	08:14	Regular	Domestico	Despegue	EZE	BAR	AEROLINEAS ARGENTINAS SA	BO-737-800	81
10	2021-01-01	08:35	Regular	Domestico	Despegue	EZE	USU	AEROLINEAS ARGENTINAS SA	BO-8737-8MB	74

Detalle de aeropuerto_detalle tabla



The screenshot shows a PostgreSQL query window with the following SQL query:

```
select * from aeropuerto_detalle_tabla limit 10;
```

The results are displayed in a table with 14 columns: id, tipo, denominacion, coordenadas, latitud, longitud, elev, uom_elev, ref, distancia_ref, direccion_ref, condicion, and control. The table contains 10 rows of data.

	id	tipo	denominacion	coordenadas	latitud	longitud	elev	uom_elev	ref	distancia_ref	direccion_ref	condicion	control
1	1	Aeródromo	CORONEL BOGADO/AGROSERVICIOS	33°16'20"S 60°34'14"W	-60.57066000	-33.27226000	44	Metros	Coronel Bogado	6	NE	PRIVADO	NOCONTROL
2	2	Aeródromo	GENERAL ACHA	37°24'6"S 64°36'49"W	-64.61351000	-37.40164000	277	Metros	General Acha	3	SO	PUBLICO	NOCONTROL
3	3	Aeródromo	ARRICIFES/LA CURA MALAL	34°4'33"S 60°8'30"W	-60.14170000	-34.07574000	37	Metros	Arrecifes	4	OSO	PRIVADO	NOCONTROL
4	4	PUD	PUERTO DESEADO	47°44'6"S 65°54'15"W	-65.90410000	-47.73511000	82	Metros	Puerto Deseado	2	N	PUBLICO	AERADIO
5	5	Aeródromo	BANDERA/AGROSERVICIOS DOÑA TERESA	28°51'19"S 62°15'53"W	-62.26462000	-28.85541000	75	Metros	Bandera	4	N	PRIVADO	NOCONTROL
6	6	Aeródromo	BANDERA/AGROSERVICIOS DOÑA TERESA	28°52'11"S 62°14'11"W	-62.23812000	-28.86691000	87	Metros	Bandera	3	NE	PRIVADO	NOCONTROL
7	7	AEP	BUENOS AIRES/AEROPORQUE J. NEWBERY	34°33'32"S 58°24'59"W	-58.41638889	-34.55888889	5.6	Metros	Ciudad de Buenos Aires	2	NE	PUBLICO	CONTROL
8	8	Aeródromo	PIEDRA DEL AGUILA	40°11'32"S 70°0'39"W	-70.01075000	-40.19232000	649	Metros	Piedra del Aguilá	17	SSE	PRIVADO	NOCONTROL
9	9	Aeródromo	ALTA GRACIA	31°39'4"S 64°23'38"W	-64.39388889	-31.65111111	533	Metros	Alta Gracia	2	E	PUBLICO	NOCONTROL
10	10	Aeródromo	CHACABUCO/LAS DOS A	34°48'40"S 60°30'60"W	-60.51661000	-34.81116000	60	Metros	Chacabuco	20	SSO	PRIVADO	NOCONTROL

```
SELECT COUNT(at2.hora_utc) AS cantidad_vuelos
from aeropuerto_tabla at2
WHERE at2.fecha BETWEEN "2021-12-01" AND "2022-01-31";
```

```
--Punto 6
SELECT COUNT(at2.hora_utc) AS cantidad_vuelos
FROM aeropuerto_tabla at2
WHERE at2.fecha BETWEEN "2021-12-01" AND "2022-01-31";
```

aeropuerto_tabla * <localhost> Script-15 x

--Punto 6

```
SELECT COUNT(at2.hora_utc) AS cantidad_vuelos
from aeropuerto_tabla at2
WHERE at2.fecha BETWEEN "2021-12-01" AND "2022-01-31";
```

Resultados 1 x

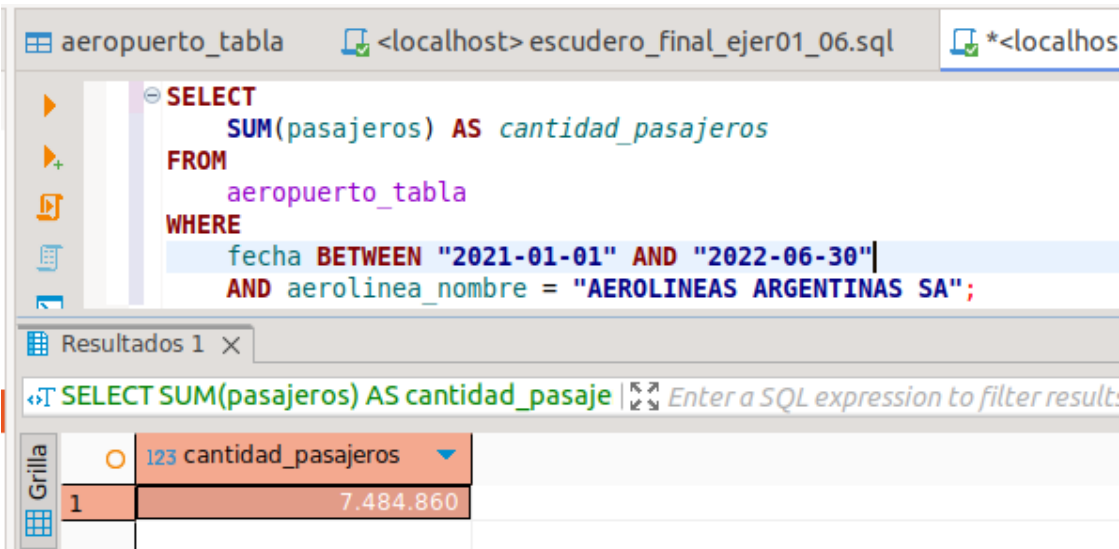
SELECT COUNT(at2.hora_utc) AS cantidad_vuelos | Enter a SQL expression to filter results

Grilla		123 cantidad_vuelos
1		57.984

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

```
SELECT
    SUM(pasajeros) AS cantidad_pasajeros
FROM
    aeropuerto_tabla
WHERE
    fecha BETWEEN "2021-01-01" AND "2022-06-30"
    AND aerolinea_nombre = "AEROLINEAS ARGENTINAS SA";
```

8.-



The screenshot shows a SQL query execution window. The query is the same as in the previous block. The results pane shows a single row with the value 7.484.860 for the column 'cantidad_pasajeros'.

cantidad_pasajeros
7.484.860

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 01/01/2022y el 30/06/2022 ordenados por fecha de manera descendiente. Mostrar consulta y Resultado de la query
--eje 01 pto 8

```
SELECT
    fecha,
    hora_utc,
    aeropuerto AS codigo_aeropuerto_salida,
    origen_destino AS codigo_aeropuerto_arribo,
    pasajeros
FROM
    aeropuerto_tabla
WHERE
    fecha BETWEEN '2022-01-01' AND '2022-06-30'
ORDER BY
    fecha DESC;
```


-- eje 01 pto 8

```

SELECT
  fecha,
  hora_utc,
  aeropuerto AS codigo_aeropuerto_salida,
  origen_destino AS codigo_aeropuerto_arribo,
  pasajeros
FROM
  aeropuerto_tabla
WHERE
  fecha BETWEEN '2022-01-01' AND '2022-06-30'
ORDER BY
  fecha DESC;

```

Resultados 1

SELECT fecha, hora_utc, aeropuerto AS codigo_aeropuerto_salida, origen_destino AS codigo_aeropuerto_arribo, pasajeros

	fecha	hora_utc	codigo_aeropuerto_salida	codigo_aeropuerto_arribo	pasajeros
1	2022-06-30	23:55	AER	TUC	0
2	2022-06-30	23:53	CBA	NEU	0
3	2022-06-30	23:49	DOZ	CBA	0
4	2022-06-30	23:48	AER	CBA	0
5	2022-06-30	23:48	CRV	AER	0
6	2022-06-30	23:47	CBA	AER	90
7	2022-06-30	23:47	AER	PAR	42
8	2022-06-30	23:45	IGU	AER	82
9	2022-06-30	23:41	MDP	FDO	0
10	2022-06-30	23:39	AER	SAL	87
11	2022-06-30	23:39	ROS	SAL	3
12	2022-06-30	23:38	NEU	AER	0
13	2022-06-30	23:38	AER	DOZ	0
14	2022-06-30	23:37	ROS	AER	42
15	2022-06-30	23:37	AER	BAR	0

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 Query

-- EJE 01 PTO 9

```

SELECT
  aerolinea_nombre,
  SUM(pasajeros) AS total_pasajeros
FROM
  aeropuerto_tabla
WHERE
  fecha BETWEEN '2021-01-01' AND '2022-06-30'
  AND aerolinea_nombre IS NOT NULL --EVITAMOS LAS AEROLINEAS SIN NOMBRE
  AND aerolinea_nombre != '0' --EVITAMOS LAS AEROLINEAS SIN NOMBRE
GROUP BY
  aerolinea_nombre
ORDER BY
  total_pasajeros DESC
LIMIT 10;

```

vuelosdb aeronaves aeropuertos_baires <localhost> Script-27 <localhost>

```

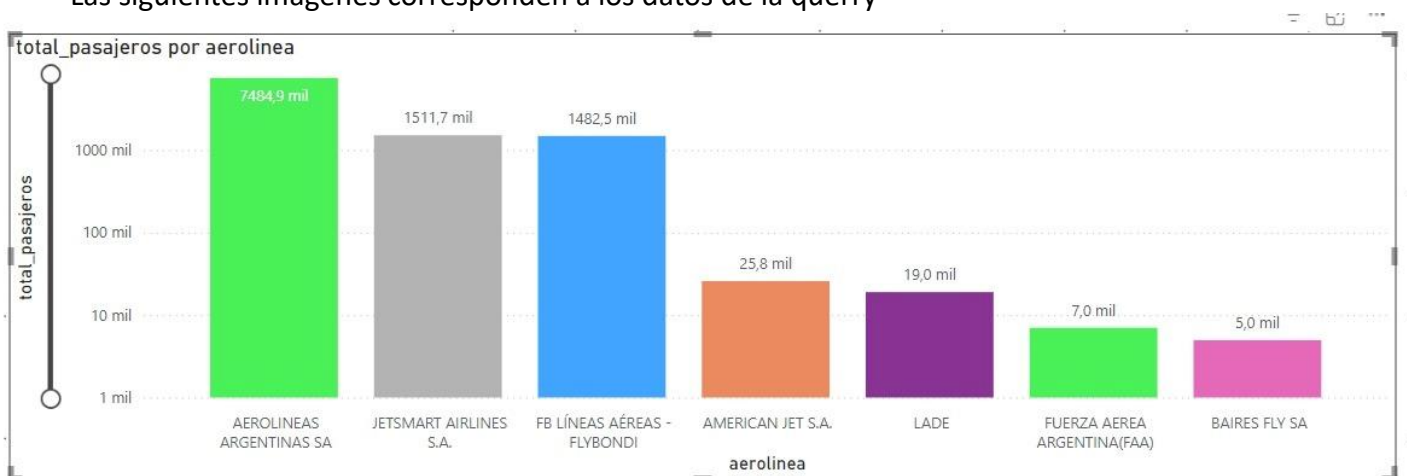
-- EJE 01 PTO 9
SELECT
  aerolinea_nombre,
  SUM(pasajeros) AS total_pasajeros
FROM
  aeropuerto_tabla
WHERE
  fecha BETWEEN '2021-01-01' AND '2022-06-30'
  AND aerolinea_nombre IS NOT NULL --EVITAMOS LAS AEROLINEAS SIN NOMBRE
  AND aerolinea_nombre != '0'--EVITAMOS LAS AEROLINEAS SIN NOMBRE
GROUP BY
  aerolinea_nombre
ORDER BY
  total_pasajeros DESC
LIMIT 10;
  
```

Resultados 1 X

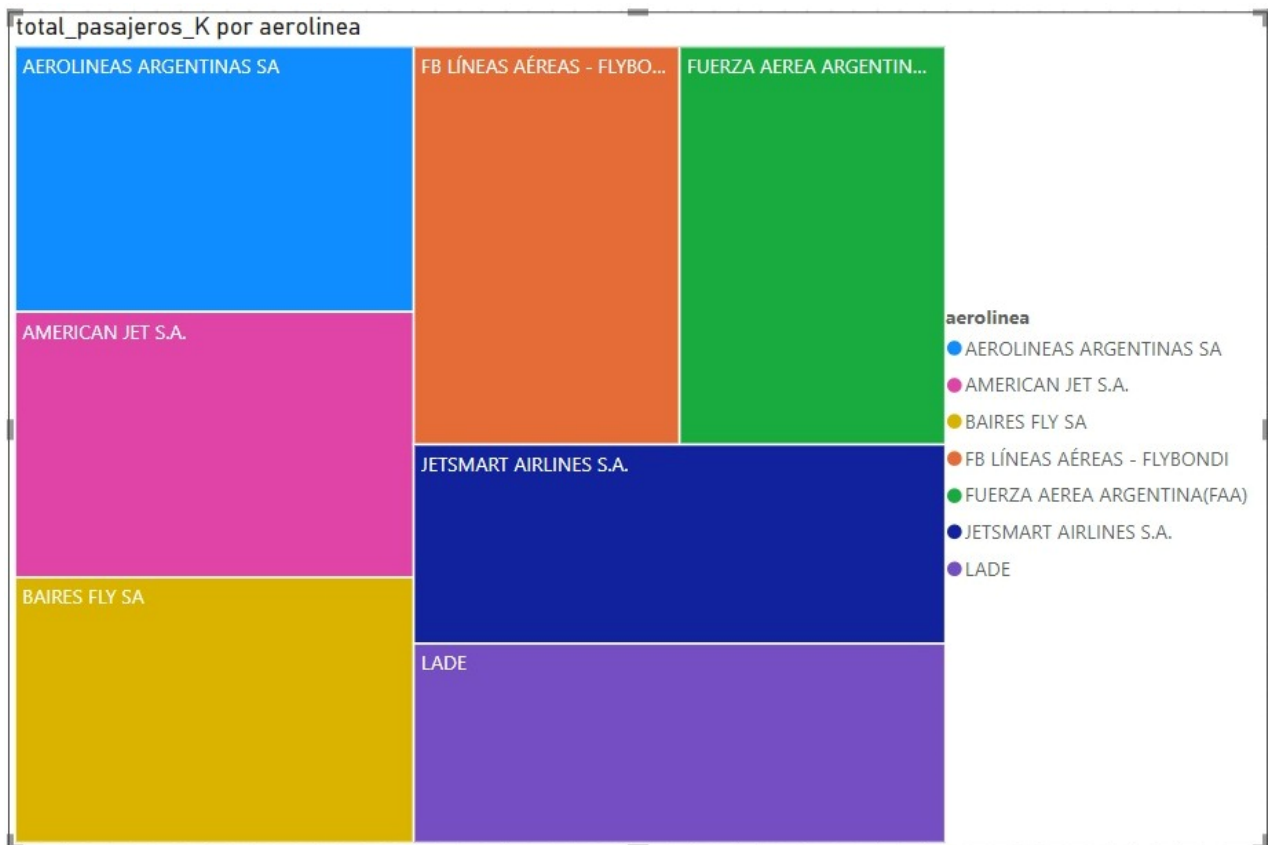
SELECT aerolinea_nombre, SUM(pasajeros) AS total_pasajeros | Enter a SQL expression to filter res

	A-Z aerolinea_nombre	123 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

Estos datos los importamos y graficamos con PowerBI.
 Las siguientes imágenes corresponden a los datos de la query



-- Gráfico de barras con índice deslizable para poder ver las aerolíneas que tiene menos pasajeros.



En ambas gráficas vemos que las tres primeras aerolíneas por cantidad total de pasajeros son:

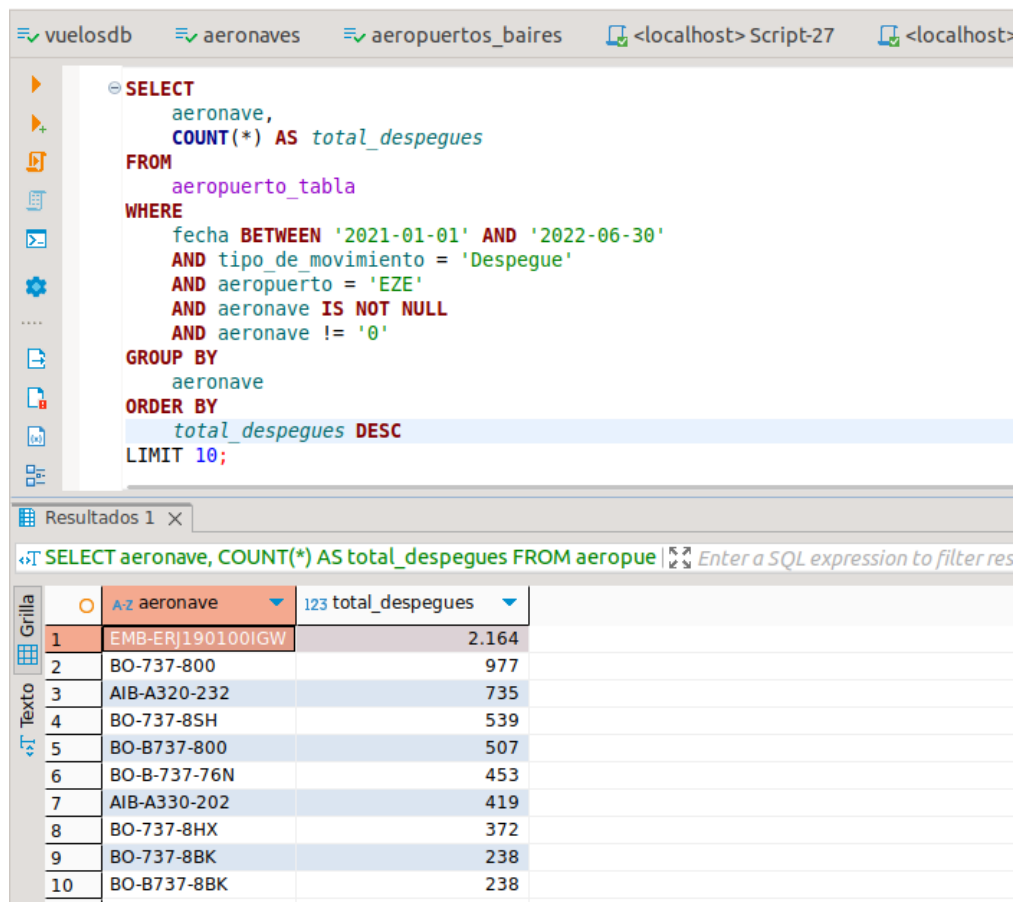
- 1.- Aerolíneas Argentinas
- 2.- Jetsmart Airline
- 3.- FlyBondi

Siendo Aerolíneas Argentinas la primera en cantidad de pasajeros.

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
    aeronave,
    COUNT(*) AS total_despegues
FROM
    aeropuerto_tabla
WHERE
    fecha BETWEEN '2021-01-01' AND '2022-06-30'
    AND tipo_de_movimiento = 'Despegue'
    AND aeropuerto = 'EZE'
    AND aeronave IS NOT NULL
    AND aeronave != '0'
GROUP BY
    aeronave
```

ORDER BY
total_despegues **DESC**
LIMIT 10;



The screenshot shows a SQL IDE with a query editor and a results pane. The query is as follows:

```
SELECT
    aeronave,
    COUNT(*) AS total_despegues
FROM
    aeropuerto_tabla
WHERE
    fecha BETWEEN '2021-01-01' AND '2022-06-30'
    AND tipo_de_movimiento = 'Despegue'
    AND aeropuerto = 'EZE'
    AND aeronave IS NOT NULL
    AND aeronave != '0'
GROUP BY
    aeronave
ORDER BY
    total_despegues DESC
LIMIT 10;
```

The results pane shows the following data:

	Az aeronave	total_despegues
1	EMB-ERJ190100IGW	2.164
2	BO-737-800	977
3	AIB-A320-232	735
4	BO-737-85H	539
5	BO-B737-800	507
6	BO-B-737-76N	453
7	AIB-A330-202	419
8	BO-737-8HX	372
9	BO-737-8BK	238
10	BO-B737-8BK	238

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

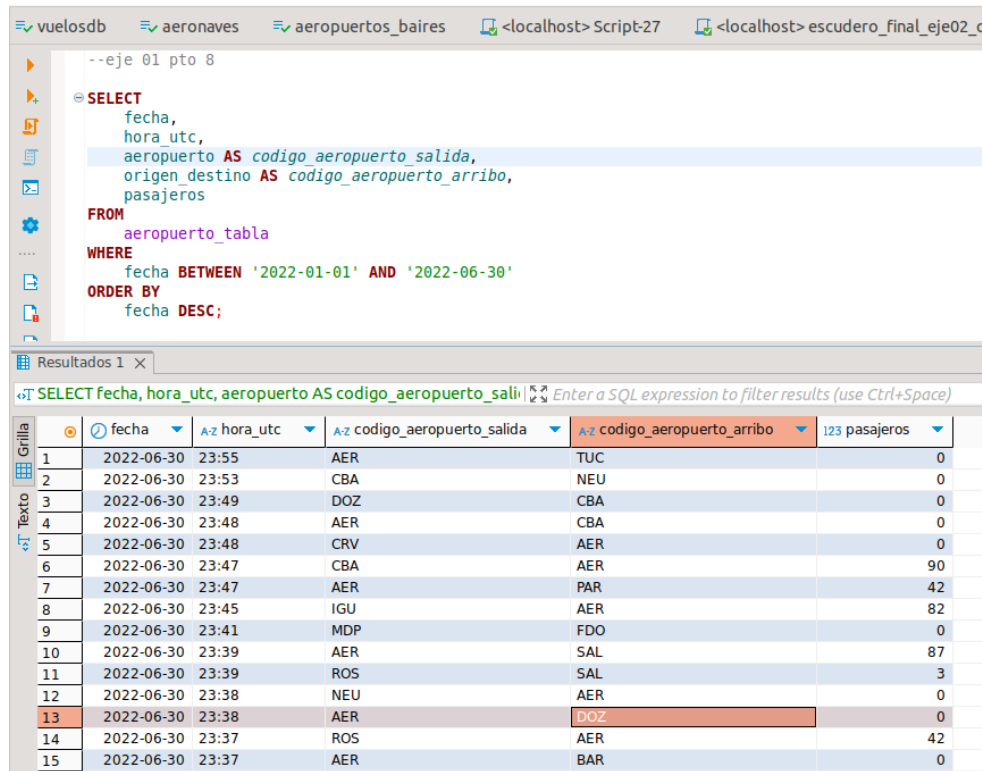
Los datos externos que se podrían agregar son:

- 1.- datos meteorológicos
- 2.- datos de cancelaciones, llegadas on-time y tiempos de vuelo para evaluar la calidad de las aerolíneas.
- 3.- datos sobre el tipo de aeronave, capacidad de pasajeros,parado por mantenimiento, autonomía, consumo de combustible para hacer una evaluación de las aeronaves más eficientes o confiables o el cálculo de emisiones de CO2 por vuelo.

4.- datos sobre las tarifas para identificar las rutas más competitivas según aeronave o época del año.

12. Elabore sus conclusiones y recomendaciones sobre este proyecto.

De la consulta 8



The screenshot shows a database query interface with a SQL query editor and a results table. The query is as follows:

```
-- eje 01 pto 8
SELECT
    fecha,
    hora_utc,
    aeropuerto AS codigo_aeropuerto_salida,
    origen_destino AS codigo_aeropuerto_arribo,
    pasajeros
FROM
    aeropuerto_tabla
WHERE
    fecha BETWEEN '2022-01-01' AND '2022-06-30'
ORDER BY
    fecha DESC;
```

The results table, titled 'Resultados 1', displays the following data:

	fecha	hora_utc	AZ codigo_aeropuerto_salida	AZ codigo_aeropuerto_arribo	123 pasajeros
1	2022-06-30	23:55	AER	TUC	0
2	2022-06-30	23:53	CBA	NEU	0
3	2022-06-30	23:49	DOZ	CBA	0
4	2022-06-30	23:48	AER	CBA	0
5	2022-06-30	23:48	CRV	AER	0
6	2022-06-30	23:47	CBA	AER	90
7	2022-06-30	23:47	AER	PAR	42
8	2022-06-30	23:45	IGU	AER	82
9	2022-06-30	23:41	MDP	FDO	0
10	2022-06-30	23:39	AER	SAL	87
11	2022-06-30	23:39	ROS	SAL	3
12	2022-06-30	23:38	NEU	AER	0
13	2022-06-30	23:38	AER	DOZ	0
14	2022-06-30	23:37	ROS	AER	42
15	2022-06-30	23:37	AER	BAR	0

Se puede analizar los patrones de vuelo, las rutas más habituales y el volumen de pasajeros y podemos contestar preguntas sobre

¿Qué aeropuerto tuvo más vuelos de salida durante este período?

¿Qué rutas tienen el mayor número de pasajeros?

¿Cuántos vuelos llegaron a un aeropuerto específico?

También se podría proponer de patrones horarios o frecuencia de vuelos para los vuelos ya que contamos con la fecha y hora de los mismos.

También se puede agregar una análisis de las rutas con baja ocupación y ver alguna mejora sobre la misma.

Por ejemplo a través de los datos de la consulta del punto 9

vuelosdb aeronaves aeropuertos_baires <localhost> Script-27 <localhost>																																			
<pre> -- EJE 01 PTO 9 SELECT aerolinea_nombre, SUM(pasajeros) AS total_pasajeros FROM aeropuerto_tabla WHERE fecha BETWEEN '2021-01-01' AND '2022-06-30' AND aerolinea_nombre IS NOT NULL --EVITAMOS LAS AEROLINEAS SIN NOMBRE AND aerolinea_nombre != '0'--EVITAMOS LAS AEROLINEAS SIN NOMBRE GROUP BY aerolinea_nombre ORDER BY total_pasajeros DESC LIMIT 10; </pre>																																			
Resultados 1 X																																			
SELECT aerolinea_nombre, SUM(pasajeros) AS total_pasajeros Enter a SQL expression to filter res																																			
Grilla	<table> <thead> <tr> <th></th><th>A-Z aerolinea_nombre</th><th>123 total_pasajeros</th></tr> </thead> <tbody> <tr><td>1</td><td>AEROLINEAS ARGENTINAS SA</td><td>7.484.860</td></tr> <tr><td>2</td><td>JETSMART AIRLINES S.A.</td><td>1.511.650</td></tr> <tr><td>3</td><td>FB LINEAS AEREAS - FLYBONDI</td><td>1.482.473</td></tr> <tr><td>4</td><td>AMERICAN JET S.A.</td><td>25.789</td></tr> <tr><td>5</td><td>L.A.D.E.</td><td>15.074</td></tr> <tr><td>6</td><td>BAIRES FLY SA</td><td>4.960</td></tr> <tr><td>7</td><td>LADE</td><td>3.895</td></tr> <tr><td>8</td><td>FUERZA AEREA ARGENTINA</td><td>3.855</td></tr> <tr><td>9</td><td>FUERZA AEREA ARGENTINA (FAA)</td><td>3.138</td></tr> <tr><td>10</td><td>FLYING AMERICA SA</td><td>2.839</td></tr> </tbody> </table>			A-Z aerolinea_nombre	123 total_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	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
	A-Z aerolinea_nombre	123 total_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	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																																	
Texto																																			

lo que vemos es que Aerolíneas Argentina tiene la mayor cantidad de pasajeros, una análisis adicional que se podría hacer es debido a qué tiene la mayor cantidad de pasajeros:

¿ tiene más rutas, más aviones, más horarios, más destinos, más personal?

Se pueden comprar las aeronaves de una compañía vs otra y analizar las rutas y la cantidad de pasajeros y, si se agrega los datos externos de las aeronaves, se podría analizar la eficiencia de los vuelos entre compañías.

Con los datos actuales se podrían analizar otras cuestiones y se enriquece el análisis con otras tablas como las mostradas se pueden incluir análisis de eficiencia en los vuelos, análisis de sostenibilidad calculando las emisiones asociadas las variabilidad de las tarifas de cada compañía vs rutas, número de pasajeros, época del año.

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

Para un análisis más comercial o de optimización por parte de algunas de las compañías aéreas, quizás es más interesante y dinámico la implementación a través de alguna plataforma CLOUD tipo GCP.

Este tipo de plataformas permite escalar los recursos de ser necesario, de forma modular, además de ofrecer todo un abanico de servicios de seguridad y permisos sobre el manejo de los datos los cuales, en otro tipo de implementación, llevaría muchos recursos técnicos y humanos.

A continuación describo el proceso a través de GCP:

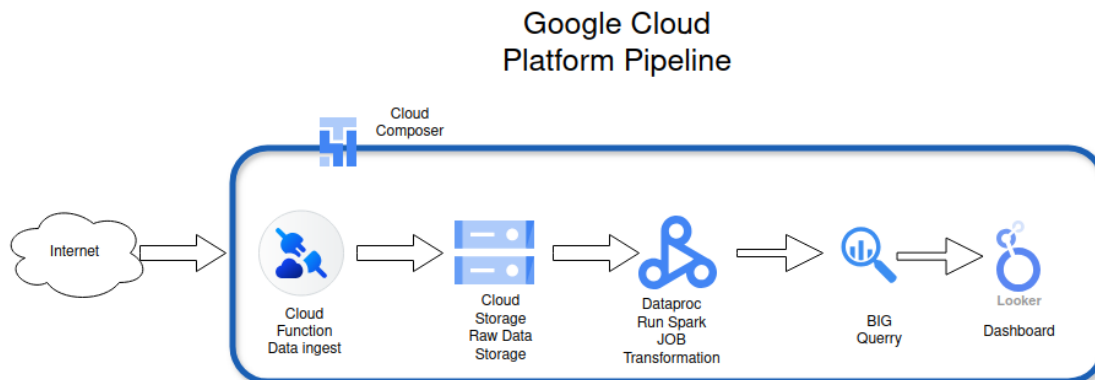
1.- Data Ingestion, la ingestión de datos se puede realizar a través de Cloud function realizando un scripts similar al data-ingest.sh realizado en el DAG sobre Airflow. Estos datos los podemos almacenar en Google Cloud y tenerlos disponibles para su procesamiento posterior.

2.- Data Transformation, a través de Dataproc podemos correr el script de transformación realizado vuelos_transformacion.py y automatizarlos.

3.- **Orchestration**, en el ecosistema de GCP podemos utilizar Cloud composer, quien utiliza Airflow sobre GCP, para realizar la orquestación similar al DAG en Airflow.

4.- Database, una vez transformados los datos, los mismos los podemos almacenar en Big Query, el motor de base de datos de GCP para su posterior análisis según lo solicitado en el ejercicio, hacer las distintas querries. Una vez realizada las querries, las mismas se puede graficar, por ejemplo con Looker o con Power BI, dos herramientas para realizar gráficas que se conectan a Big query y permite realizar análisis gráficos sobre los datos. Estos análisis se mantienen actualizados al estar todo el pipeline automatizado a través de las distintos servicios disponibles en Google Cloud Platform.

A continuación podrán ver una imagen del pipeline de este ejercicio realizado para GCP:



--