

Examen Final

Alumno: Federico Bonino

Fecha: 24 de Junio de 2024

Ejercicio 1 - Aviación Civil

Consigna

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.

Desarrollo

1. Para realizar la ingesta se utiliza el siguiente script ubicado en `/home/hadoop/scripsts/ingest_aviacion.sh`

```
rm -f /home/hadoop/landing/*.  
  
wget -O /home/hadoop/landing/2021-informe-ministerio.csv  
"https://edvaibucket.blob.core.windows.net/data-engineer-edvai/2021-  
informe-ministerio.csv?sp=r&st=2023-11-06T12:59:46Z&se=2025-11-  
06T20:59:46Z&sv=2022-11-02&sr=b&sig=%2BSs5xIW3qcwmRh5TTmheIY9ZBa9BJC8XQDcI  
%2FPLRe9Y%3D"  
  
wget -O /home/hadoop/landing/2022-informe-ministerio.csv  
"https://edvaibucket.blob.core.windows.net/data-engineer-edvai/202206-  
informe-ministerio.csv?sp=r&st=2023-11-06T12:52:39Z&se=2025-11-  
06T20:52:39Z&sv=2022-11-  
02&sr=c&sig=J4Ddi2c7Ep230hQLPisbYaerlH472iigPwc1%2FkG80EM%3D"  
  
wget -O /home/hadoop/landing/aeropuertos_detalle.csv  
"https://edvaibucket.blob.core.windows.net/data-engineer-edvai/aeropuertos  
_detalle.csv?sp=r&st=2023-11-06T12:52:39Z&se=2025-11-06T20:52:39Z&sv=2022-  
11-02&sr=c&sig=J4Ddi2c7Ep230hQLPisbYaerlH472iigPwc1%2FkG80EM%3D"  
  
/home/hadoop/hadoop/bin/hdfs dfs -rm /ingest/*.  
  
/home/hadoop/hadoop/bin/hdfs dfs -put /home/hadoop/landing/*. /ingest
```

2. Para crear las tablas en el datawarehouse se utiliza el siguiente script ubicado en

/home/hadoop/scripts/create_database.hql

El script se corrió manualmente por consola antes de realizar la ingesta mediante

hive -f /home/hadoop/scripts/create_database.hql

```
-- Create the aviacion_civil database
create database aviacion_civil;
use aviacion_civil;

-- create tables with appropriate schema
create table
    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)
row format delimited
fields terminated by ',';

create 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 ',';
```

3. Para orquestar el proceso de ETL mediante Airflow se utiliza el siguiente script ubicado en `/home/hadoop/airflow/dags/dag_aviacion_civil.py`

```
from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.empty import EmptyOperator
from datetime import datetime, timedelta

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

with DAG(
    'aviacion_civil_elt',
    default_args=args,
    schedule_interval='@once',
    start_date=datetime(2020, 1, 1),
    dagrun_timeout=timedelta(minutes=60),
    tags=['ingest', 'transform']
)as dag:

    ingest = BashOperator(
        task_id='ingest',
        # Don't drop the space at the end of the command or Jinja will
        fail
        bash_command='/usr/bin/sh /home/hadoop/scripts/ingest_aviacion.sh
    ',
    )

    transform = BashOperator(
        task_id='transform',
        # Don't drop the space at the end of the command or Jinja will
        fail
        bash_command='ssh hadoop@172.17.0.2 /home/hadoop/spark/bin/spark-submit --files /home/hadoop/hive/conf/hive-site.xml /home/hadoop/scripts/transform_aviacion.py ',
    )

    end_processing = EmptyOperator(
        task_id='end_processing',
    )

    ingest >> transform >> end_processing
```

Además en el proyecto se incluyen los archivos

Esta captura de pantalla muestra la ejecución exitosa del pipeline orquestado por Airflow. El entorno completo se ejecuta de manera local containerizado en Docker y se accede a las web Uis mediante el navegador (en los puertos correspondientes expuestos) y a los CLI desde la consola con `docker exec -it edvai_hadoop bash`

The screenshot displays the Airflow web interface for a DAG named 'aviacion_civil_elt'. The interface is accessed via a browser at `localhost:8010/dags/aviacion_civil_elt/graph`. The top navigation bar includes links for DAGs, Security, Browse, Admin, and Docs, along with a clock showing 01:37 UTC and a user profile icon labeled 'AA'.

The DAG header shows the name 'DAG: aviacion_civil_elt' and its current state 'queued'. It also indicates the schedule '@once' and the next run time 'None'.

Below the header is a toolbar with various views: Grid, Graph (selected), Calendar, Task Duration, Task Tries, Landing Times, Gantt, Details, Code, and Audit Log. There are also buttons for 'Grid', 'Graph', 'Calendar', 'Task Duration', 'Task Tries', 'Landing Times', 'Gantt', 'Details', 'Code', and 'Audit Log'.

The main area of the interface shows the DAG graph. At the top, there are filters for 'Runs' (25) and a 'Run' dropdown set to 'scheduled__2020-01-01T00:00:00+00:00'. There is also a 'Layout' dropdown set to 'Left > Right' and an 'Update' button. Below these is a search bar labeled 'Find Task...'. A legend shows various task statuses: queued, running, success, failed, up_for_retry, up_for_reschedule, upstream_failed, skipped, scheduled, deferred, and no_status. The DAG graph itself consists of three tasks: 'ingest', 'transform', and 'end_processing', connected in a linear sequence.

4. Para orquestar el realizar las transformaciones en PySpark se utiliza el siguiente script ubicado en `/home/hadoop/scripts/transform_aviacion.py`

```

from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
sc = SparkContext('local')
spark = SparkSession(sc)
from pyspark.sql.functions import *
from pyspark.sql import HiveContext
hc = HiveContext(sc)

# Leemos los csv desde HDFS y cargamos en dataframes
df1 = spark.read.option("header", "true").option("sep", ";")\
    .csv("hdfs://172.17.0.2:9000/ingest/2021-informe-ministerio.csv")
df2 = spark.read.option("header", "true").option("sep", ";")\
    .csv("hdfs://172.17.0.2:9000/ingest/2022-informe-ministerio.csv")
df_aerop = spark.read.option("header", "true").option("sep", ";")\
    .csv("hdfs://172.17.0.2:9000/ingest/aeropuertos_detalle.csv")

# Unimos las tablas de vuelos
df_vuelos = df1.union(df2).distinct()

# Dropeamos columnas que no utilizaremos
df_vuelos = df_vuelos.drop('Calidad dato')
df_aerop = df_aerop.drop("inhab", "fir")

# Normalizamos nombres de columnas
df_vuelos = df_vuelos.withColumnRenamed("Hora UTC", "horaUTC")\
    .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")
df_aerop = df_aerop.withColumnRenamed("local", "aeropuerto")\
    .withColumnRenamed("oaci", "oac")

# Casteamos las variables no-string
df_vuelos = df_vuelos\
    .withColumn("fecha", to_date("fecha", "dd/MM/yyyy").alias("fecha"))\
    .withColumn("pasajeros", col('pasajeros').cast('int'))
df_aerop = df_aerop.withColumn("elev", col('elev').cast('float'))\
    .withColumn("distancia_ref", col('distancia_ref').cast('float'))

# Filtramos por fecha
df_vuelos = df_vuelos.filter("fecha ≥ '2021-01-01' AND fecha ≤ '2022-06-30'")

# Dropeamos los vuelos internacionales
df_vuelos = df_vuelos.filter("clasificacion_de_vuelo ≠ 'Internacional'")

```

```

# Tratamos los nulos
df_vuelos = df_vuelos.withColumn('pasajeros',
    when(col('pasajeros').isNull(), 0 ).otherwise(col('pasajeros')))
df_aerop = df_aerop.withColumn('distancia_ref',
    when(col('distancia_ref').isNull(), 0. )
    .otherwise(col('distancia_ref')))

# Creamos vistas con la data filtrada
df_vuelos.createOrReplaceTempView("df_vuelos_vista")
df_aerop.createOrReplaceTempView("df_aerop_vista")

# Insertamos DFs filtrados en tablas de Hive
hc.sql("insert into aviacion_civil.aeropuerto_tabla select * from
df_vuelos_vista;")
hc.sql("insert into aviacion_civil.aeropuerto_detalle_tabla select * from
df_aerop_vista;")

```

5. Chequeamos que el esquema de las tablas sea el correcto, con sus nombres y tipos. Las capturas de pantalla fueron realizadas una vez corrido con éxito el proceso de orquestación en Airflow y los datos procesados por PySpark y ya cargados en Hive.

```

hive> describe formatted 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:           aviacion_civil
Owner:              hadoop
CreateTime:         Sat Jun 22 04:53:14 ART 2024
LastAccessTime:     UNKNOWN
Retention:          0
Location:           hdfs://172.17.0.2:9000/user/hive/warehouse/aviacion_civil.db/aeropuerto_tabla
Table Type:         MANAGED_TABLE
Table Parameters:
    numFiles         1
    numRows          0
    rawDataSize       0
    totalSize         42121415
    transient_lastDdlTime 1719115281

# Storage Information
SerDe Library:      org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:        org.apache.hadoop.mapred.TextInputFormat
OutputFormat:       org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:         No
Num Buckets:        -1
Bucket Columns:     []
Sort Columns:       []
Storage Desc Params:
    field.delim      ,
    serialization.format ,
Time taken: 0.136 seconds, Fetched: 39 row(s)

```

```

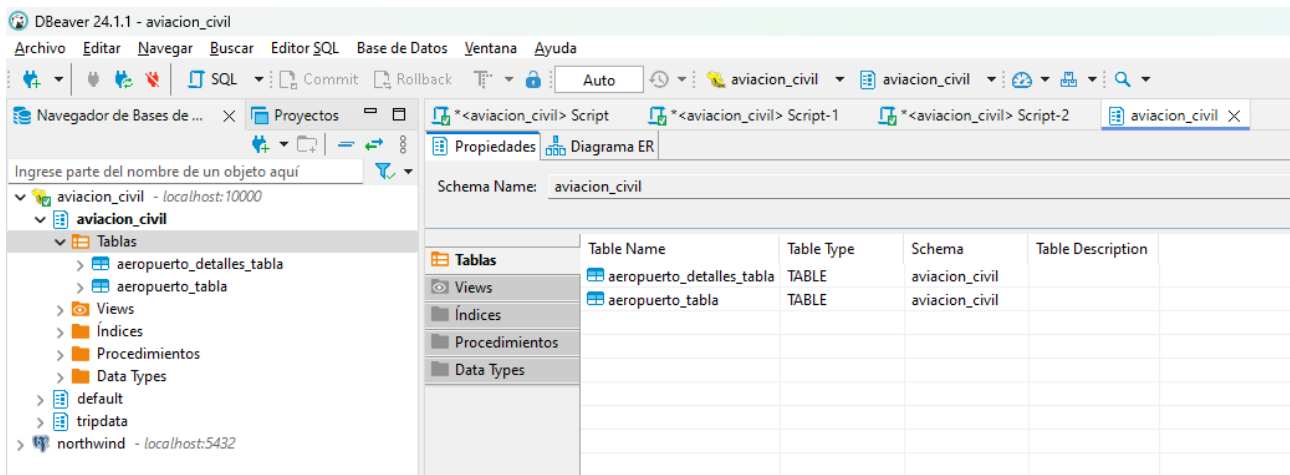
hive> describe formatted aeropuerto_detallestable;
OK
# col_name          data_type          comment
aeropuerto          string
oac                  string
iata                 string
tipo                string
denominacion         string
coordenadas          string
latitud              string
longitud             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:            aviacion_civil
Owner:                hadoop
CreateTime:          Sat Jun 22 04:53:15 ART 2024
LastAccessTime:      UNKNOWN
Retention:            0
Location:             hdfs://172.17.0.2:9000/user/hive/warehouse/aviacion_civil.db/aeropuerto_detallestable
Table Type:          MANAGED_TABLE
Table Parameters:
    numFiles          1
    numRows            0
    rawDataSize        0
    totalSize          129700
    transient_lastDdlTime 1719115283

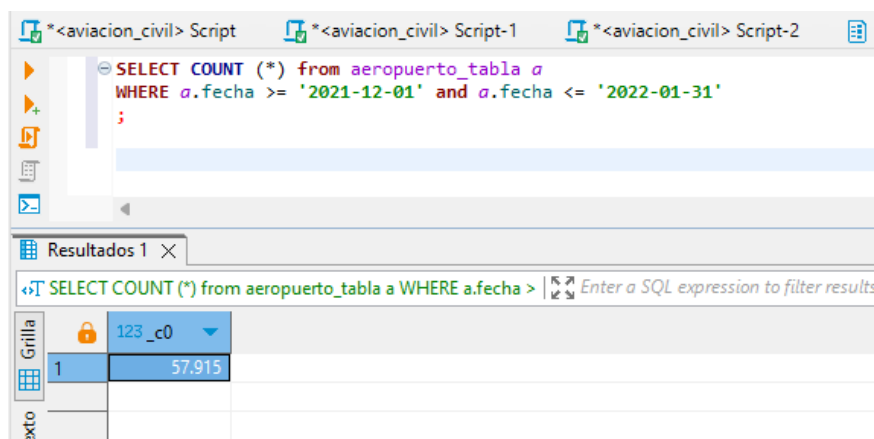
# Storage Information
SerDe Library:        org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:          org.apache.hadoop.mapred.TextInputFormat
OutputFormat:         org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:           No
Num Buckets:          -1
Bucket Columns:       []
Sort Columns:         []
Storage Desc Params:  field.delim ,

```

Para la siguiente sección realizo las consultas a Hive desde Dbeaver por una cuestión de comodidad y practicidad de la herramienta. Para ello conectamos con Hive en el puerto 10000:



6. Entre las fechas 01/12/2021 y 31/01/2022 se realizaron 57915 vuelos.



A modo de ejemplo se muestra también la captura de la misma query realizada en la consola de Hive:

```
hive> select count(*) from aeropuerto_tabla as a where a.fecha>='2021-12-01' AND a.fecha<='2022-01-31'
Automatically selecting local only mode for query
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider
Query ID = hadoop_20240623042313_a1b5880e-6437-4f9b-ab8a-b5425ea28558
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2024-06-23 04:23:15,762 Stage-1 map = 0%, reduce = 0%
2024-06-23 04:23:16,771 Stage-1 map = 100%, reduce = 0%
2024-06-23 04:23:17,777 Stage-1 map = 100%, reduce = 100%
Ended Job = job_local829094355_0003
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 168991238 HDFS Write: 523 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
57915
Time taken: 4.439 seconds, Fetched: 1 row(s)
```


7. Entre el 01/01/2021 y el 30/06/2022 viajaron 7.483.736 de pasajeros por Aerolíneas Argentinas.

Script

```
SELECT SUM(a.pasajeros) from aeropuerto_tabla a
WHERE a.fecha >= '2021-01-01' and a.fecha <= '2022-06-30'
and a.aerolinea_nombre == 'AEROLINEAS ARGENTINAS SA'
```

Resultados 1

SELECT SUM(a.pasajeros) from aeropuerto_tabla a WHERE a.f

Grilla	123_c0
1	7.483.736

8. 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.
Total = 95671 registros.

Script

```
SELECT
a.fecha, a.horaUTC, a.aeropuerto as origen,
adt.`ref` as ciudad_origen, a.origen_destino as destino,
adt2.`ref` as ciudad_destino, a.pasajeros
FROM aeropuerto_tabla as a
left join aeropuerto_detalle as adt
on a.aeropuerto == adt.aeropuerto
left join aeropuerto_detalle as adt2
on a.origen_destino == adt2.aeropuerto
WHERE a.fecha >= '2022-01-01' and a.fecha <= '2022-06-30'
and a.tipo_de_movimiento == 'Despegue'
ORDER by a.fecha desc, a.horaUTC desc
```

Resultados 1

SELECT a.fecha, a.horaUTC, a.aeropuerto as origen, adt.`ref` as ciudad_

Grilla	fecha	horaUTC	ABC origen	ABC ciudad_origen	ABC destino	ABC ciudad_destino	123 pasajeros
1	2022-06-30	23:59	DOZ	Mendoza	AER	Ciudad de Buenos Aires	0
2	2022-06-30	23:49	DOZ	Mendoza	CBA	Córdoba	0
3	2022-06-30	23:48	AER	Ciudad de Buenos Aires	CBA	Córdoba	0
4	2022-06-30	23:47	CBA	Córdoba	AER	Ciudad de Buenos Aires	90
5	2022-06-30	23:39	AER	Ciudad de Buenos Aires	SAL	Salta	87
6	2022-06-30	23:37	AER	Ciudad de Buenos Aires	BAR	San Carlos de Bariloche	0
7	2022-06-30	23:37	ROS	Rosario	AER	Ciudad de Buenos Aires	42
8	2022-06-30	23:35	GAL	Río Gallegos	AER	Ciudad de Buenos Aires	0
9	2022-06-30	23:30	AER	Ciudad de Buenos Aires	SDE	Santiago del Estero	86
10	2022-06-30	23:28	BAR	San Carlos de Bariloche	AER	Ciudad de Buenos Aires	0
11	2022-06-30	23:25	OSA	Santa Rosa	EZE	Capital Federal	1
12	2022-06-30	23:22	SAL	Salta	EZE	Capital Federal	41
13	2022-06-30	23:22	AER	Ciudad de Buenos Aires	BAR	San Carlos de Bariloche	0
14	2022-06-30	23:21	IGU	Cataratas del Iguazú	EZE	Capital Federal	37
15	2022-06-30	23:20	SRA	San Rafael	SRA	San Rafael	0
16	2022-06-30	23:20	TUC	San Miguel de Tucumán	AER	Ciudad de Buenos Aires	85

9. Las 10 aerolíneas que más pasajeros llevaron entre el 01/01/2021 y el 30/06/2022 exceptuando aquellas aerolíneas que no tengan nombre.

```

SELECT
    SUM(a.pasajeros) as total_pasajeros,
    a.aerolinea_nombre
FROM aeropuerto_tabla as a
WHERE a.fecha >= '2021-01-01' and a.fecha <= '2022-06-30'
    and a.aerolinea_nombre != '0'
GROUP by a.aerolinea_nombre
SORT BY total_pasajeros desc
LIMIT 10
;

```

	123 total_pasajeros	ABC aerolinea_nombre
1	7.483.736	AEROLINEAS ARGENTINAS SA
2	1.511.567	JETSMART AIRLINES S.A.
3	1.482.290	FB LINEAS AÉREAS - FLYBONDI
4	25.789	AMERICAN JET S.A.
5	15.074	L.A.D.E.
6	4.960	BAIRES FLY SA
7	3.895	LADE
8	3.855	FUERZA AEREA ARGENTINA
9	3.138	FUERZA AEREA ARGENTINA (FAA)
10	2.839	FLYING AMERICA SA

10. 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.

```

SELECT
    COUNT(*) as total_vuelos,
    at2.aeronave
FROM (
    SELECT * from aeropuerto_tabla as a
    WHERE a.fecha >= '2021-01-01' and a.fecha <= '2022-06-30'
        and (a.aeropuerto == 'EZE' or a.aeropuerto == 'AER')
        and a.aeronave != '0'
) as at2
GROUP by at2.aeronave
SORT BY total_vuelos desc
LIMIT 10
;

```

	123 total_vuelos	ABC aeronave
1	22.365	EMB-ERJ190100IGW
2	10.504	AIB-A320-232
3	8.734	BO-737-800
4	5.234	BO-B737-800
5	4.020	BO-737-8
6	3.923	BO-737-8Q8
7	3.412	BO-737-8SH
8	2.997	BO-B-737-76N
9	2.708	BO-737-8HX
10	2.636	BO-B737-8SH

11. Considero que contar con un identificador único de vuelo facilitaría grandemente las consultas sobre la base.

12. _

13. _