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.

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

```
ruta='/home/hadoop/landing/'
ruta_hdfs='/home/hadoop/hadoop/bin/'

rm -f "${ruta}"*.*

wget -P "${ruta}" https://dataengineerpublic.blob.core.windows.net/data-engineer/2021-informe-ministerio.csv

wget -P "${ruta}" https://dataengineerpublic.blob.core.windows.net/data-engineer/202206-informe-ministerio.csv

wget -P "${ruta}" https://dataengineerpublic.blob.core.windows.net/data-engineer/aeropuertos_detalle.csv

"${ruta_hdfs}hdfs" dfs -rm /ingest/*.*

"${ruta_hdfs}hdfs" dfs -put "${ruta}"*.* /ingest
```

```
hadoop@ec27db0d59e9:~/scripts$ hdfs dfs -ls /ingest
Found 3 items
-rw-r--r--   1 hadoop supergroup   32322556 2024-06-17 20:00 /ingest/2021-informe-ministerio.csv
-rw-r--r--   1 hadoop supergroup   22833520 2024-06-17 20:00 /ingest/202206-informe-ministerio.csv
-rw-r--r--   1 hadoop supergroup     136007 2024-06-17 20:00 /ingest/aeropuertos_detalle.csv
hadoop@ec27db0d59e9:~/scripts$
```

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)**

```
hive> create database aviacion;
OK
Time taken: 0.545 seconds
```

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

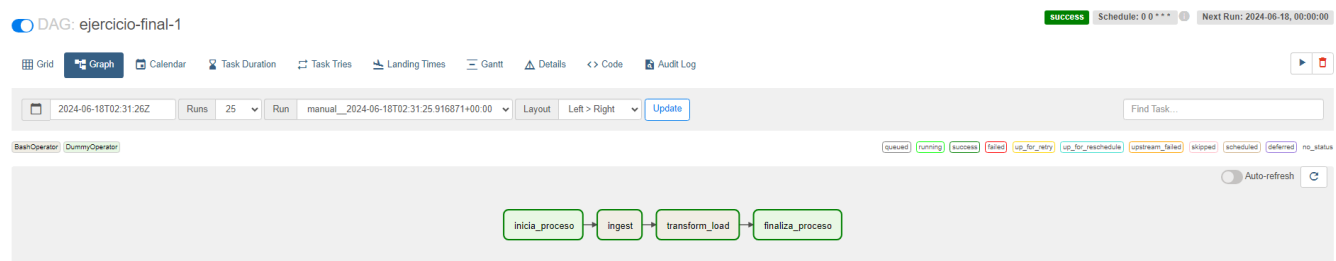
```
CREATE EXTERNAL TABLE aviacion.aeropuerto_tabla(fecha date, horaUTC string, clase_de_vuelo string, clasificacion_de_vuelo string, tipo_de_movimiento string, aeropuerto string, origen_destino string, aerolinea_nombre string, aeronave string, pasajeros int)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LOCATION '/tables/external/aviacion/aeropuerto_tabla';

CREATE EXTERNAL TABLE aviacion.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 ','
LOCATION '/tables/external/aviacion/aeropuerto_detalle_tabla';
```

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_detalles_tabla

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

```
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql import HiveContext
from pyspark.sql import functions as F
from pyspark.sql.functions import col, to_date
sc = SparkContext('local')
spark = SparkSession(sc)
hc = HiveContext(sc)

##leo csv desde HDFS y lo cargo en un dataframe
df = 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/202206-informe-ministerio.csv")
df3 = spark.read.option("header", "true").option("sep", ";",).csv("hdfs://172.17.0.2:9000/ingest/aeropuertos_detalle.csv")

##elimino espacios y caracteres en los headers
df = df.select([F.col(x).alias(x.replace(' ', '').replace('ó', 'o').replace('/', '').replace('(', '').replace(')', '')) for x in df.columns])
df = df.withColumn('fecha', to_date(col('Fecha'), 'd/M/yyyy'))
df2 = df2.select([F.col(x).alias(x.replace(' ', '').replace('ó', 'o').replace('/', '').replace('(', '').replace(')', '')) for x in df2.columns])
df2 = df2.withColumn('fecha', to_date(col('Fecha'), 'd/M/yyyy'))

##creamos vistas de los df
df.createOrReplaceTempView("2021_informe")
df2.createOrReplaceTempView("202206_informe")
df3.createOrReplaceTempView("aeropuertos")

##filtramos los df
df_2021=  spark.sql("select fecha, HoraUTC as horautc, ClasedeVuelotodoslosvuelos as clase_de_vuelo, ClasificacionVuelo as clasificacion_de_vuelo, TipodeMovimiento as tipo_de_movimiento, Aeropuerto as aeropuerto,
OrigenDestino as origen_destino, AerolineaNombre as aerolinea_nombre, Aeronave as aeronave, COALESCE(CAST(Pasajeros AS int),0) as pasajeros from 2021_informe where ClasificacionVuelo = 'Domestico'")

df_202206=  spark.sql("select fecha, HoraUTC as horautc, ClasedeVuelotodoslosvuelos as clase_de_vuelo, ClasificacionVuelo as clasificacion_de_vuelo, TipodeMovimiento as tipo_de_movimiento, Aeropuerto as aeropuerto,
OrigenDestino as origen_destino, AerolineaNombre as aerolinea_nombre, Aeronave as aeronave, COALESCE(CAST(Pasajeros AS int),0) as pasajeros from 202206_informe where ClasificacionVuelo = 'Doméstico'")

df_2021.createOrReplaceTempView("2021_informe_cast")
df_202206.createOrReplaceTempView("202206_informe_cast")

df_union =  spark.sql("select * from 2021_informe_cast union select * from 202206_informe_cast")

df_union.createOrReplaceTempView("informes_union")

df_final = spark.sql("select * from informes_union where fecha between '2021-01-01' and '2022-06-30'")

df_final.createOrReplaceTempView("informes_tabla_final")
```

```
df_aeropuertos =  spark.sql("select local as aeropuerto, oaci as oac, iata, tipo, denominacion, coordenadas, latitud, longitud, CAST(elev AS float), uom_elev, ref, COALESCE(CAST(distancia_ref AS float),0) as
distancia_ref, direccion_ref, condicion, control, region, uso, trafico,sna, concesionado, provincia from aeropuertos")

df_aeropuertos.createOrReplaceTempView("aeropuertos_tabla_final")

##insertamos los df en Hive
spark.sql("insert into aviacion.aeropuerto_tabla select * from informes_tabla_final")

spark.sql("insert into aviacion.aeropuerto_detalle_tabla select * from aeropuertos_tabla_final")
```



```python
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='ejercicio-final-1',
    default_args=args,
    schedule_interval='0 0 * * *',
    start_date=days_ago(2),
    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='ingest',
        bash_command='/usr/bin/sh /home/hadoop/scripts/ingest_final_1.sh ',
    )
    transform_load = BashOperator(
        task_id='transform_load',
        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_ejercicio_final_1.py ',
    )



    inicia_proceso >> ingest >> transform_load >> finaliza_proceso
```

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.)**

```
>>> df_final.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)

>>>
```

```
>>> df_aeropuertos.printSchema()
root
 |-- 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)

>>>
```

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

```
select count(*) as cantidad_vuelos
from aeropuerto_tabla at
where fecha BETWEEN '2021-12-01' and '2022-01-31'
```

	cantidad_vuelos
1	57,915

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 at
where aerolinea_nombre = 'AEROLINEAS ARGENTINAS SA' and fecha BETWEEN '2021-01-01' and '2022-06-30'
```

	cantidad_pasajeros
1	7,483,736

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**

```
select at.fecha, at.horautc , at.aeropuerto, adt_origen.ref as aeropuerto_ciudad, at.origen_destino, adt_salida.ref as origen_destino_ciudad, SUM(at.pasajeros) as cantidad_pasajeros
from aeropuerto_tabla at
left join aeropuerto_detalle_tabla adt_origen on at.aeropuerto = adt_origen.aeropuerto
left join aeropuerto_detalle_tabla adt_salida on at.origen_destino  = adt_salida.aeropuerto
where at.aerolinea_nombre = 'AEROLINEAS ARGENTINAS SA' and at.fecha BETWEEN '2022-01-01' and '2022-06-30'
GROUP by at.fecha, at.horautc , at.aeropuerto, at.origen_destino,adt_salida.ref,adt_origen.ref
order by at.fecha DESC
```

	fecha	horautc	aeropuerto	aeropuerto_ciudad	origen_destino	origen_destino_ciudad	cantidad_pasajeros
1	2022-06-30	23:47	AER	Ciudad de Buenos Aires	PAR	Paraná	42
2	2022-06-30	23:45	IGU	Cataratas del Iguazú	AER	Ciudad de Buenos Aires	82
3	2022-06-30	23:39	AER	Ciudad de Buenos Aires	SAL	Salta	87
4	2022-06-30	23:38	NEU	Neuquén	AER	Ciudad de Buenos Aires	0
5	2022-06-30	23:37	ROS	Rosario	AER	Ciudad de Buenos Aires	42
6	2022-06-30	23:35	AER	Ciudad de Buenos Aires	CBA	Córdoba	0
7	2022-06-30	23:30	AER	Ciudad de Buenos Aires	SDE	Santiago del Estero	86
8	2022-06-30	23:48	CRV	Comodoro Rivadavia	AER	Ciudad de Buenos Aires	0
9	2022-06-30	23:28	BAR	San Carlos de Bariloche	AER	Ciudad de Buenos Aires	0
10	2022-06-30	23:26	AER	Ciudad de Buenos Aires	IGU	Cataratas del Iguazú	0
11	2022-06-30	23:59	ECA	El Calafate	AER	Ciudad de Buenos Aires	0
12	2022-06-30	23:22	SAL	Salta	EZE	Capital Federal	41
13	2022-06-30	23:21	IGU	Cataratas del Iguazú	EZE	Capital Federal	37
14	2022-06-30	23:20	TUC	San Miguel de Tucumán	AER	Ciudad de Buenos Aires	85
15	2022-06-30	23:19	SAL	Salta	AER	Ciudad de Buenos Aires	0
16	2022-06-30	23:13	PAR	Paraná	AER	Ciudad de Buenos Aires	42
17	2022-06-30	23:13	EZE	Capital Federal	FSA	Formosa	54
18	2022-06-30	23:12	AER	Ciudad de Buenos Aires	MDP	Mar del Plata	35

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

```
select aerolinea_nombre, SUM(pasajeros) as cantidad_pasajeros
from aeropuerto_tabla at
where aerolinea_nombre NOT LIKE '0' and fecha BETWEEN '2021-01-01' and '2022-06-30'
group by aerolinea_nombre
order by cantidad_pasajeros DESC
limit 10
```

	aerolinea_nombre	cantidad_pasajeros
1	AEROLINEAS ARGENTINAS SA	7,483,736
2	JETSMART AIRLINES S.A.	1,511,567
3	FB LÍNEAS AÉREAS - FLYBONDI	1,482,290
4	AMERICAN JET S.A.	25,789
5	L.A.D.E.	15,074
6	BAIRES FLY SA	4,960
7	LADE	3,895
8	FUERZA AEREA ARGENTINA	3,855
9	FUERZA AEREA ARGENTINA (FAA)	3,138
10	FLYING AMERICA SA	2,839

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

```
select at.aeronave, COUNT(at.aeronave) as cantidad
from aeropuerto_tabla at
left join aeropuerto_detalle_tabla adt on at.aeropuerto = adt.aeropuerto
where at.aeronave NOT LIKE '0'
and at.tipo_de_movimiento = 'Despegue'
and (adt.provincia  = 'BUENOS AIRES' or adt.provincia  = 'CIUDAD AUTÓNOMA DE BUENOS AIRES')
and at.fecha BETWEEN '2021-01-01' and '2022-06-30'
group by at.aeronave
order by cantidad DESC
limit 10
```

	aeronave	cantidad
1	EMB-ERJ190100IGW	12,470
2	CE-150-L	8,090
3	CE-152	7,927
4	CE-150-M	6,058
5	AIB-A320-232	5,345
6	BO-737-800	4,535
7	CE-150-J	3,008
8	CE-150-G	2,866
9	BO-B737-800	2,735
10	PA-PA-28-181	2,453

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

Se podría agregar a los informes anuales datos que determinen si los vuelos se realizan días feriados o especiales.
También, si los despegues o aterrizajes son realizados en la hora estipulada y si es posible determinando algún motivo, climático, mecánico, etc.

12. Elabore sus conclusiones y recomendaciones sobre este proyecto.

Conclusiones:
- Aerolíneas Argentinas es la empresa más elegida para realizar viajes de cabotaje. A través de una encuesta de satisfacción se podría determinar por qué se elige AA.
- Embraer EMB-ERJ190100IGW es el avión más utilizado para viajes de cabotaje. Se podría verificar si se debe a que cubren las rutas más demandadas, costo de mantenimiento, etc.

Recomendaciones:
-Impedir que se carguen datos nulos o no se carguen datos para todas las clases de vuelo. Tanto para los campos aeronaves y aerolíneas_nombre podría distorsionar el resultado obtenido.

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