

# 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/lv/dataset/transporte-aterrizajes-despegues-procesados-por-administracion-nacional-aviacion-civil-anac>

Listado de detalles de aeropuertos de Argentina:

<https://datos.transporte.gob.ar/dataset/lista-aeropuertos>

## TAREAS

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

2021:

<https://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=%2BSs5xIW3qcwmRh5TTmheIY9ZBa9BJC8XQDcl%2FPLRe9Y%3D>

2022:

<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=J4Ddi2c7Ep23OhQLPisbYaerIH472iigPwc1%2FkG80EM%3D>

Aeropuertos\_detalle:

[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=J4Ddi2c7Ep23OhQLPisbYaerIH472iigPwc1%2FkG80EM%3D](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=J4Ddi2c7Ep23OhQLPisbYaerIH472iigPwc1%2FkG80EM%3D)

```
#!/bin/bash

# Clean landing directory and get data
rm /home/hadoop/landing/*

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"

# Clean HDFS directory and put the data from landing into Hadoop
/home/hadoop/hadoop/bin/hdfs dfs -rm /ingest/*
/home/hadoop/hadoop/bin/hdfs dfs -put /home/hadoop/landing/* /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 (aeropuertos\_detalle.csv)

Schema Tabla 1:

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

```
create database administracionNacionalDeAviacionCivil;

use administracionNacionalDeAviacionCivil;

create external 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 ','
-- location '/tables/external/tripdata';
;

describe formatted aeropuerto_tabla;
```

```
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	

Schema Tabla 2:

Campo Tipo
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

```
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_trafico STRING,  
    sna STRING,  
    concesionado STRING,  
    provincia STRING  
)  
-- row format delimited  
-- fields terminated by ','  
-- location '/tables/external/tripdata';  
;  
  
describe formatted aeropuerto_detalle_tabla;
```

```
Time taken: 0.231 seconds
```

```
hive> describe formatted aeropuerto_detalle_tabla;
```

```
OK
```

#	col_name	data_type	comment
1	aeropuerto	string	
2	oac	string	
3	iata	string	
4	tipo	string	
5	denominacion	string	
6	coordenadas	string	
7	latitud	string	
8	longitud	string	
9	elev	float	
10	uom_elev	string	
11	ref	string	
12	distancia_ref	float	
13	direccion_ref	string	
14	condicion	string	
15	control	string	
16	region	string	
17	uso_trafico	string	
18	sna	string	
19	concesionado	string	
20	provincia	string	

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

```
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='TP1Final',
    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:

    comienza_proceso = DummyOperator(
        task_id='comienza_proceso',
    )

    finaliza_proceso = DummyOperator(
        task_id='finaliza_proceso',
    )

    ingest = BashOperator(
        task_id='ingest',
        bash_command='/usr/bin/sh /home/hadoop/scripts/ingestFinalTP1.sh',
    )

    transform = BashOperator(
        task_id='transform',
        bash_command='ssh hadoop@172.17.0.2
```



```

/home/hadoop/spark/bin/spark-submit --files
/home/hadoop/hive/conf/hive-site.xml
/home/hadoop/scripts/sparkFinalTP1.py',
)

comienza_proceso >> ingest >> transform >> finaliza_proceso

if __name__ == "__main__":

    dag.cli()

```

#### 4. Realizar las siguientes 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.functions import col, to_date

sc = SparkContext('local')
spark = SparkSession(sc)
hc = HiveContext(sc)

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/202206-informe-ministerio.csv")
df3 = spark.read.option("header", "true").option("sep",
";").csv("hdfs://172.17.0.2:9000/ingest/aeropuertos_detalle.csv")

dfunion = df1.union(df2)

aeropuerto_tabla_todo = dfunion.select(
    to_date(dfunion["Fecha"], "dd/MM/yyyy").alias("fecha"),
    dfunion["Hora UTC"].alias("horaUTC"),

```

```

dfunion["Clase de Vuelo (todos los vuelos)"].alias("clase_de_vuelo"),
dfunion["Clasificación Vuelo"].alias("clasificacion_de_vuelo"),
dfunion["Tipo de Movimiento"].alias("tipo_de_movimiento"),
dfunion["Aeropuerto"].alias("aeropuerto"),
dfunion["Origen / Destino"].alias("origen_destino"),
dfunion["Aerolinea Nombre"].alias("aerolinea_nombre"),
dfunion["Aeronave"].alias("aeronave"),
dfunion["Pasajeros"].cast("int").alias("pasajeros")
).fillna({'pasajeros': 0})

aeropuerto_tabla = aeropuerto_tabla_todo.filter(col("fecha") < '2022-06-30')

aeropuerto_detalle = df3.select(
    col("local").alias("aeropuerto"),
    col("oaci").alias("oaci"),
    col("iata").alias("iata"),
    col("tipo").alias("tipo"),
    col("denominacion").alias("denominacion"),
    col("coordenadas").alias("coordenadas"),
    col("latitud").alias("latitud"),
    col("longitud").alias("longitud"),
    col("elev").cast("float").alias("elev"),
    col("uom_elev").alias("uom_elev"),
    col("ref").alias("ref"),
    col("distancia_ref").cast("float").alias("distancia_ref"),
    col("direccion_ref").alias("direccion_ref"),
    col("condicion").alias("condicion"),
    col("control").alias("control"),
    col("region").alias("region"),
    (col("uso") + col("trafico")).alias("uso_trafico"),
    col("sna").alias("sna"),
    col("concesionado").alias("concesionado"),
    col("provincia").alias("provincia")
).fillna({'distancia_ref': 0})

aeropuerto_tabla.write.insertInto("administracionNacionalDeAviacionCivil.aeropuerto_tabla")
aeropuerto_detalle.write.insertInto("administracionNacionalDeAviacionCivil.aeropuerto_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.)

```
>>> aeropuerto_tabla.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)
```

Time taken: 0.231 seconds

```
hive> describe formatted aeropuerto_detalle;

```

OK

#	col_name	data_type	comment
1	aeropuerto	string	
2	oac	string	
3	iata	string	
4	tipo	string	
5	denominacion	string	
6	coordenadas	string	
7	latitud	string	
8	longitud	string	
9	elev	float	
10	uom_elev	string	
11	ref	string	
12	distancia_ref	float	
13	direccion_ref	string	
14	condicion	string	
15	control	string	
16	region	string	
17	uso_trafico	string	
18	sna	string	
19	concesionado	string	
20	provincia	string	

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

```
SELECT
COUNT(aeronave) AS cantidad_de_vuelos
FROM
administracionnacionaldeaviacioncivil.aeropuerto_tabla
WHERE
fecha >= '2021-01-01'
AND fecha <= '2022-01-31';
```

Results 1 X

SELECT COUNT(aeronave) AS cantidad\_de\_vuelos | Enter a SQL expression to filter results (use Ctrl+Sp

	123 cantidad de vuelos
1	716,420

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_de_pasajeros
FROM
administracionnacionaldeaviacioncivil.aeropuerto_tabla
WHERE
fecha >= '2021-01-01'
AND fecha <= '2022-06-30'
AND aerolinea nombre ='AEROLINEAS ARGENTINAS SA';
```

Results 1 X

SELECT SUM(pasajeros) AS cantidad\_de\_pas | Enter a SQL expression to filter results (use Ctrl+Sp

	123 cantidad de pasajeros
1	17,744,844

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 ciudad_origen,
  at.origen_destino,
  adt_destino.ref AS ciudad_destino,
  at.pasajeros
FROM
  administracionnacionaldeaviacioncivil.aeropuerto_tabla at
LEFT JOIN
  administracionnacionaldeaviacioncivil.aeropuerto_detalle_tabla adt_origen ON at.aeropuerto = adt_origen.aeropuerto
LEFT JOIN
  administracionnacionaldeaviacioncivil.aeropuerto_detalle_tabla adt_destino ON at.origen_destino = adt_destino.aeropuerto
WHERE
  at.fecha >= '2022-01-01'
  AND at.fecha <= '2022-06-30';

```

	fecha	horaUTC	AER aeropuerto	AER ciudad origen	AER origen destino	AER ciudad destino	123 pasajeros
1	2022-01-01	00:01	AER	Ciudad de Buenos Aires	ECA	El Calafate	69
2	2022-01-01	00:05	AER	Ciudad de Buenos Aires	SAL	Salta	65
3	2022-01-01	00:05	IGU	Cataratas del Iguazú	AER	Ciudad de Buenos Aires	41
4	2022-01-01	00:09	AER	Ciudad de Buenos Aires	GAL	Río Gallegos	73
5	2022-01-01	00:09	EZE	Capital Federal	KDFW	[NULL]	261
6	2022-01-01	00:12	SRA	San Rafael	DOZ	Mendoza	0
7	2022-01-01	00:12	AER	Ciudad de Buenos Aires	EZE	Capital Federal	0
8	2022-01-01	00:16	AER	Ciudad de Buenos Aires	ROS	Rosario	20
9	2022-01-01	00:21	EZE	Capital Federal	AER	Ciudad de Buenos Aires	0
10	2022-01-01	00:28	ROS	Rosario	AER	Ciudad de Buenos Aires	5
11	2022-01-01	00:30	AER	Ciudad de Buenos Aires	CBA	Córdoba	39
12	2022-01-01	00:42	AER	Ciudad de Buenos Aires	CRR	Corrientes	85
13	2022-01-01	00:42	JUJ	San Salvador de Jujuy	EZE	Capital Federal	88
14	2022-01-01	00:43	EZE	Capital Federal	MMMX	[NULL]	93
15	2022-01-01	00:48	CBA	Córdoba	AER	Ciudad de Buenos Aires	85
16	2022-01-01	01:00	AER	Ciudad de Buenos Aires	ROS	Rosario	5
17	2022-01-01	01:02	EZE	Capital Federal	ECA	El Calafate	31
18	2022-01-01	01:03	EZE	Capital Federal	MMUN	[NULL]	271
19	2022-01-01	01:10	EZE	Capital Federal	KIAH	[NULL]	245
20	2022-01-01	01:10	EZE	Capital Federal	KMIA	[NULL]	226
21	2022-01-01	01:17	EZE	Capital Federal	SCEL	[NULL]	114

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

```

SELECT DISTINCT
aerolinea_nombre,
COUNT(pasajeros) OVER(PARTITION BY aerolinea_nombre) as pasajeros_transportados
FROM
administracionnacionaldeaviacioncivil.aeropuerto_tabla
WHERE
fecha >= '2022-01-01'
AND fecha <= '2022-06-30'
AND aerolinea_nombre IS NOT NULL
AND aerolinea_nombre != '0'
ORDER BY pasajeros_transportados DESC
LIMIT 10;

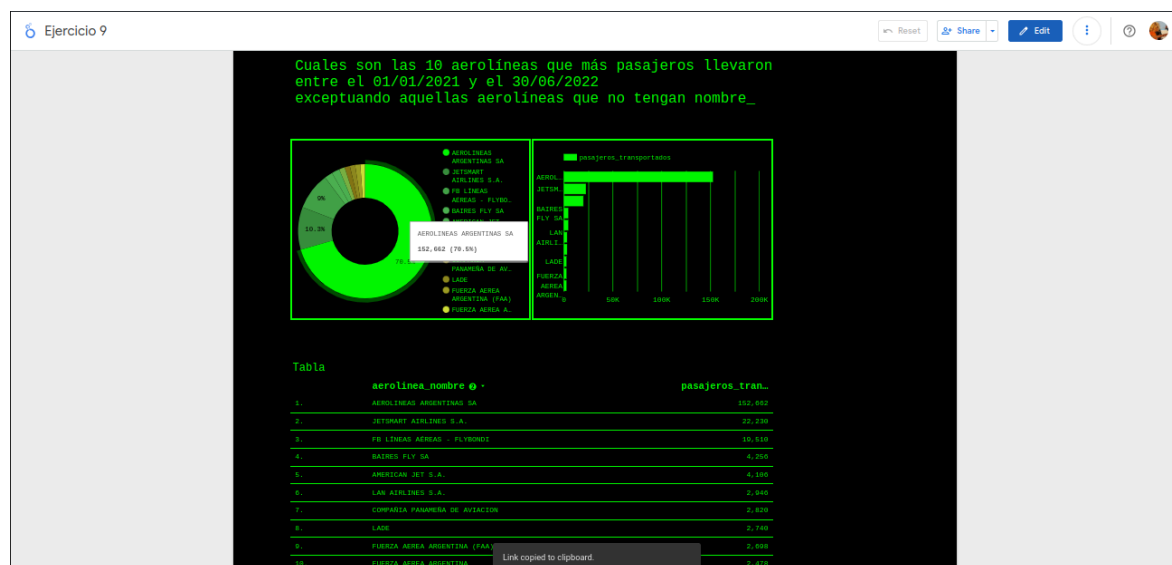
```

Results 1 ×

SELECT DISTINCT aerolinea\_nombre, COUNT(pasajeros) OVER(PARTITION BY aerolinea\_nombre) as pasajeros\_transportados

	aerolinea_nombre	pasajeros_transportados
1	AEROLINEAS ARGENTINAS SA	152,662
2	JETSMART AIRLINES S.A.	22,230
3	FB LINEAS AEREAS - FLYBONDI	19,510
4	BAIRES FLY SA	4,256
5	AMERICAN JET S.A.	4,106
6	LAN AIRLINES S.A.	2,946
7	COMPAÑIA PANAMEÑA DE AVIACION	2,820
8	LADE	2,740
9	FUERZA AEREA ARGENTINA (FAA)	2,698
10	FUERZA AEREA ARGENTINA	2,478

Link <https://lookerstudio.google.com/u/0/reporting/adcd6c78-c454-47ae-9ae2-aad2a00bf38c>



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 DISTINCT
aeronave,
COUNT(aeronave) OVER(PARTITION BY aeronave) as aeronave_recuento_de_despegues
FROM
administracionnacionaldeaviacioncivil.aeropuerto_tabla
WHERE
fecha >= '2021-01-01'
AND fecha <= '2022-06-30'
AND aeronave != '0'
AND ((aeropuerto = 'EZE' OR aeropuerto = 'AER'))
ORDER BY aeronave_recuento_de_despegues DESC
LIMIT 10;

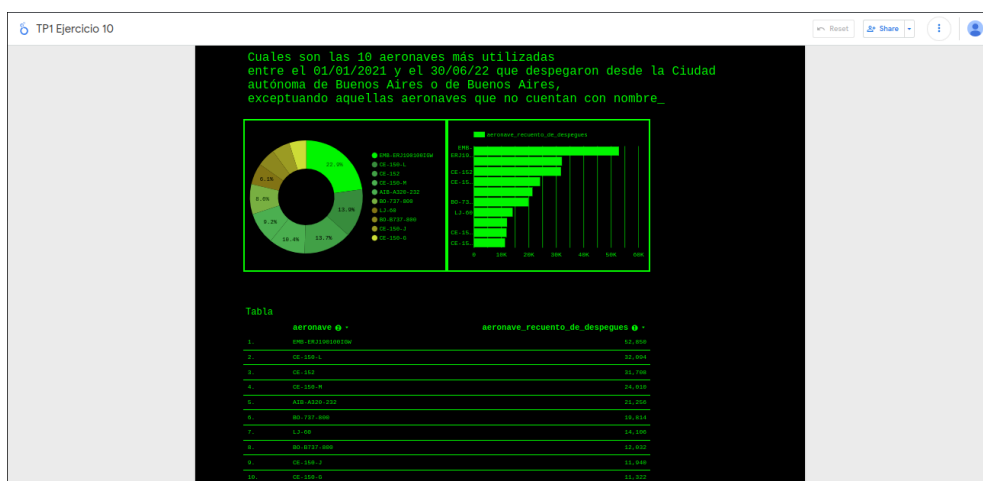
```

Results 1 x

SELECT DISTINCT aeronave, C... Enter a SQL expression to filter results (use Ctrl+Space)

	aeronave	aeronave_recuento_de_despegues
1	EMB-ERJ190100IGW	48,336
2	AIB-A320-232	21,254
3	BO-737-800	19,432
4	BO-B737-800	11,696
5	BO-737-8	8,928
6	BO-737-8Q8	8,244
7	BO-737-8SH	7,846
8	BO-B-737-76N	7,612
9	BO-737-8HX	6,006

Link <https://lookerstudio.google.com/reporting/fa67a921-af3c-43f4-ad90-9b85ea04e5c5>





Dos queries alternativas para el ej 10

```

SELECT DISTINCT
  aet.aeronave,
  COUNT(aet.aeronave) OVER(PARTITION BY aet.aeronave) as aeronave_recuento_de_despegues
FROM
  administracionnacionaldeaviacioncivil.aeropuerto_tabla aet
LEFT JOIN
  administracionnacionaldeaviacioncivil.aeropuerto_detalle_tabla adt_origen ON aet.aeropuerto = adt_
WHERE
  fecha >= '2021-01-01'
AND fecha <= '2022-06-30'
AND (adt_origen.provincia LIKE '%BUENOS AIRES%')
AND aet.aeronave != '0'
ORDER BY aeronave_recuento_de_despegues DESC
LIMIT 10;

```

Results 1 x

SELECT DISTINCT aet.aeronave; Enter a SQL expression to filter results (use Ctrl+Space)

	aeronave	aeronave recuento de despegues
1	EMB-ERJ190100IGW	52,850
2	CE-150-L	32,094
3	CE-152	31,708
4	CE-150-M	24,010
5	AIB-A320-232	21,256
6	BO-737-800	19,814
7	LJ-60	14,106
8	BO-737-800	13,873

```

SELECT DISTINCT
  aet.aeronave,
  COUNT(*) OVER(PARTITION BY aet.aeronave) as aeronave_recuento_de_despegues,
  COUNT(CASE WHEN adt_origen.provincia = 'BUENOS AIRES' THEN 1 ELSE NULL END) OVER(PARTITION BY aet.aeronave) as 'PROVINCIA DE BUENOS AIRES',
  COUNT(CASE WHEN adt_origen.provincia = 'CIUDAD AUTÓNOMA DE BUENOS AIRES' THEN 1 ELSE NULL END) OVER(PARTITION BY aet.aeronave) as 'CIUDAD DE BUENOS AIRES'
FROM
  administracionnacionaldeaviacioncivil.aeropuerto_tabla aet
LEFT JOIN
  administracionnacionaldeaviacioncivil.aeropuerto_detalle_tabla adt_origen ON aet.aeropuerto = adt_origen.aeropuerto
WHERE
  fecha >= '2021-01-01'
AND fecha <= '2022-06-30'
AND adt_origen.provincia LIKE '%BUENOS AIRES%'
AND aet.aeronave != '0'
ORDER BY
  aeronave_recuento_de_despegues DESC
LIMIT 10;

```

Results 1 x

SELECT DISTINCT aet.aeronave; Enter a SQL expression to filter results (use Ctrl+Space)

	aeronave	aeronave recuento de despegues	provincia de buenos aires	ciudad de buenos aires
1	EMB-ERJ190100IGW	52,850	13,194	39,656
2	CE-150-L	32,094	32,092	2
3	CE-152	31,708	31,708	0
4	CE-150-M	24,010	24,008	2
5	AIB-A320-232	21,256	2,648	18,608

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

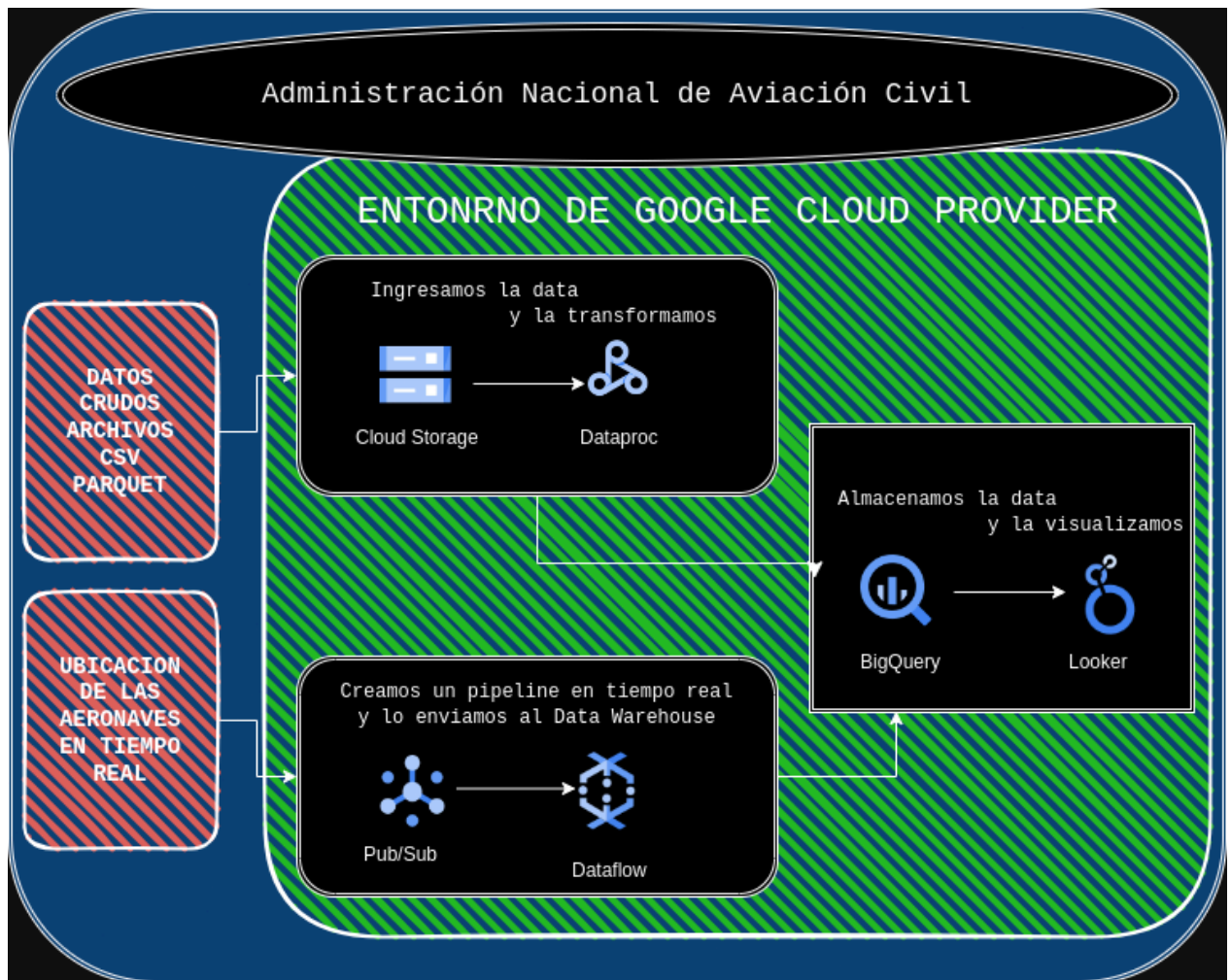
Agregaría la ubicación de las aeronaves en tiempo real. Eso permite saber si hay demoras, predecir arribos y por ende organizar mejor los aterrizajes y despegues de cada aeropuerto tanto como tener un mejor control sobre las rutas aéreas.

Y como sugerencia personal, agregaría un contacto de aquella persona que vaya a retirar al pasajero al aeropuerto (en caso que lo haya) para informar esas demoras. Lo mismo con las demoras en las partidas tener el contacto del pasajero para ahorrarle horas de espera en el aeropuerto.

12. Elabore sus conclusiones y recomendaciones sobre este proyecto.

Usaría GCP para que el proyecto sea escalable, tener mejor performance y evitar problemas con el mantenimiento de la infraestructura.

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



## Ejercicio 2:

### Alquiler de automóviles

Una de las empresas líderes en alquileres de automóviles solicita una serie de dashboards y reportes para poder basar sus decisiones en datos. Entre los indicadores mencionados se encuentran total de alquileres, segmentación por tipo de combustible, lugar, marca y modelo de automóvil, valoración de cada alquiler, etc.

Como Data Engineer debe crear y automatizar el pipeline para tener como resultado los datos listos para ser visualizados y responder las preguntas de negocio.

1. Crear en hive una database car\_rental\_db y dentro una tabla llamada car\_rental\_analytics, con estos campos:

campos tipo
fuelType string

rating integer
renterTripsTaken integer
reviewCount integer
city string
state_name string
owner_id integer
rate_daily integer

make string

model string

year integer

```
create database car_rental_db;

use car_rental_db;

create external table car_rental_analytics(
    fuelType STRING,
    rating INT,
    renterTripsTaken INT,
    reviewCount INT,
    city STRING,
    state_name STRING,
    owner_id INT,
    rate_daily INT,
    make STRING,
    model STRING,
    year INT
)
-- row format delimited
-- fields terminated by ','
-- location '/tables/external/tripdata';
;

describe formatted car_rental_analytics;
```

## 2. Crear script para el ingest de estos dos files

- <https://edvaibucket.blob.core.windows.net/data-engineer-edvai/CarRentalData.csv?sp=r&st=2023-11-06T12:52:39Z&se=2025-11-06T20:52:39Z&sv=2022-11-02&sr=c&sig=J4Ddi2c7Ep23OhQLPisbYaerIH472iigPwc1%2FkG80EM%3D>
- [https://public.opendatasoft.com/api/explore/v2.1/catalog/datasets/georef-united-states-of-america-state/exports/csv?lang=en&timezone=America%2FArgentina%2FBuenos\\_Aires&use\\_labels=true&delimiter=%3B](https://public.opendatasoft.com/api/explore/v2.1/catalog/datasets/georef-united-states-of-america-state/exports/csv?lang=en&timezone=America%2FArgentina%2FBuenos_Aires&use_labels=true&delimiter=%3B)

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

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

```
#!/bin/bash

# Clean landing directory and get data
rm /home/hadoop/landing/*

wget -P /home/hadoop/landing -O /home/hadoop/landing/CarRentalData.csv
"https://edvaibucket.blob.core.windows.net/data-engineer-edvai/CarRentalData.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 -P /home/hadoop/landing -O
/home/hadoop/landing/united-states-of-america-state-Buenos_Aires.csv
"https://public.opendatasoft.com/api/explore/v2.1/catalog/datasets/georef-united-states-of-america-state/exports/csv?lang=en&timezone=America%2FArgentina%2FBuenos_Aires&use_labels=true&delimiter=%3B"

test -f /home/hadoop/landing/CarRentalData.csv
test -f
/home/hadoop/landing//home/hadoop/landing/united-states-of-america-state-Buenos_Aires.csv

# Clean HDFS directory and put the data from landing into Hadoop
/home/hadoop/hadoop/bin/hdfs dfs -rm /ingest/*
/home/hadoop/hadoop/bin/hdfs dfs -put /home/hadoop/landing/* /ingest
```

3. Crear un script para tomar el archivo desde HDFS y hacer las siguientes transformaciones:

- En donde sea necesario, modificar los nombres de las columnas. Evitar espacios y puntos (reemplazar por \_). Evitar nombres de columna largos
- Redondear los float de 'rating' y castear a int.
- Joinear ambos files
- Eliminar los registros con rating nulo
- Cambiar mayúsculas por minúsculas en 'fuelType'
- Excluir el estado Texas

Finalmente insertar en Hive el resultado

```
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql import HiveContext

sc = SparkContext('local')
spark = SparkSession(sc)
hc = HiveContext(sc)

df1 = spark.read.option("header",
"true").csv("hdfs://172.17.0.2:9000/ingest/CarRentalData.csv")
# df2 = spark.read.option("header", "true").option("sep",
";").csv("hdfs://172.17.0.2:9000/ingest/united-states-of-america-state-Buenos_Ai
res.csv")

df1.createOrReplaceTempView("CarRentalData")

car_rental_analytics = spark.sql("""
    SELECT
        LOWER(CAST(fuelType AS STRING)) AS fuelType,
        ROUND(CAST(rating AS INT)) AS rating,
        CAST(renterTripsTaken AS INT) AS renterTripsTaken,
        CAST(reviewCount AS INT) AS reviewCount,
        CAST(`location.city` AS STRING) AS city,
        CAST(`location.state` AS STRING) AS state_name,
        CAST(`owner.id` AS INT) AS owner_id,
        CAST(`rate.daily` AS INT) AS rate_daily,
        CAST(`vehicle.make` AS STRING) AS make,
        CAST(`vehicle.model` AS STRING) AS model,
        CAST(`vehicle.year` AS INT) AS year
    FROM
        CarRentalData
    WHERE
        `location.state` != 'Texas'
""")

car_rental_analytics.write.insertInto("car_rental_db.car_rental_analytics")
```

4. Realizar un proceso automático en Airflow que orqueste los pipelines creados en los puntos anteriores. Crear dos tareas:

a. Un DAG padre que ingente los archivos y luego llame al DAG hijo

```
from datetime import timedelta
from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.dummy import DummyOperator
from airflow.operators.trigger_dagrun import TriggerDagRunOperator
from airflow.utils.dates import days_ago

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

with DAG(
    dag_id='TP2FinalPadre',
    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:

    comienza_proceso = DummyOperator(
        task_id='comienza_proceso',
    )

    finaliza_proceso = DummyOperator(
        task_id='finaliza_proceso',
    )

    ingest = BashOperator(
        task_id='ingest',
        bash_command='/usr/bin/sh /home/hadoop/scripts/ingestFinalTP2.sh ',
    )

    transform = TriggerDagRunOperator(
        task_id='trigger_target',
        trigger_dag_id='TP2FinalHijo',
        execution_date='{{ ds }}',
        reset_dag_run=True,
        wait_for_completion=True,
        poke_interval=30
    )

    comienza_proceso >> ingest >> transform >> finaliza_proceso

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



## b. Un DAG hijo que procese la información y la cargue en Hive

```

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='TP2FinalHijo',
    default_args=args,
    schedule_interval='0 0 * * *',
    start_date=days_ago(2),
    dagrun_timeout=timedelta(minutes=60),
    tags=['transform'],
    params={"example_key": "example_value"},
) as dag:

    comienza_proceso = DummyOperator(
        task_id='comienza_proceso',
    )

    finaliza_proceso = DummyOperator(
        task_id='finaliza_proceso',
    )

    transform = BashOperator(
        task_id='transform',
        bash_command='ssh hadoop@172.17.0.2 /home/hadoop/spark/bin/spark-submit --files /home/hadoop/hive/conf/hive-site.xml /home/hadoop/scripts/sparkFinalTP2.py',
    )

    comienza_proceso >> transform >> finaliza_proceso

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

```

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

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

```
SELECT
  fueltype,
  rating
FROM
  car_rental_db.car_rental_analytics
WHERE rating <= 4
AND (fueltype = 'hybrid' OR fueltype = 'electric');
SELECT
  COUNT(fueltype) AS consulta_a
FROM
  car_rental_db.car_rental_analytics
WHERE
  rating <= 4
AND (fueltype = 'hybrid' OR fueltype = 'electric');
```

Results 1 Results 2 X

SELECT COUNT(fueltype) AS c Enter a SQL expression to filter results (us

123 consulta a
335

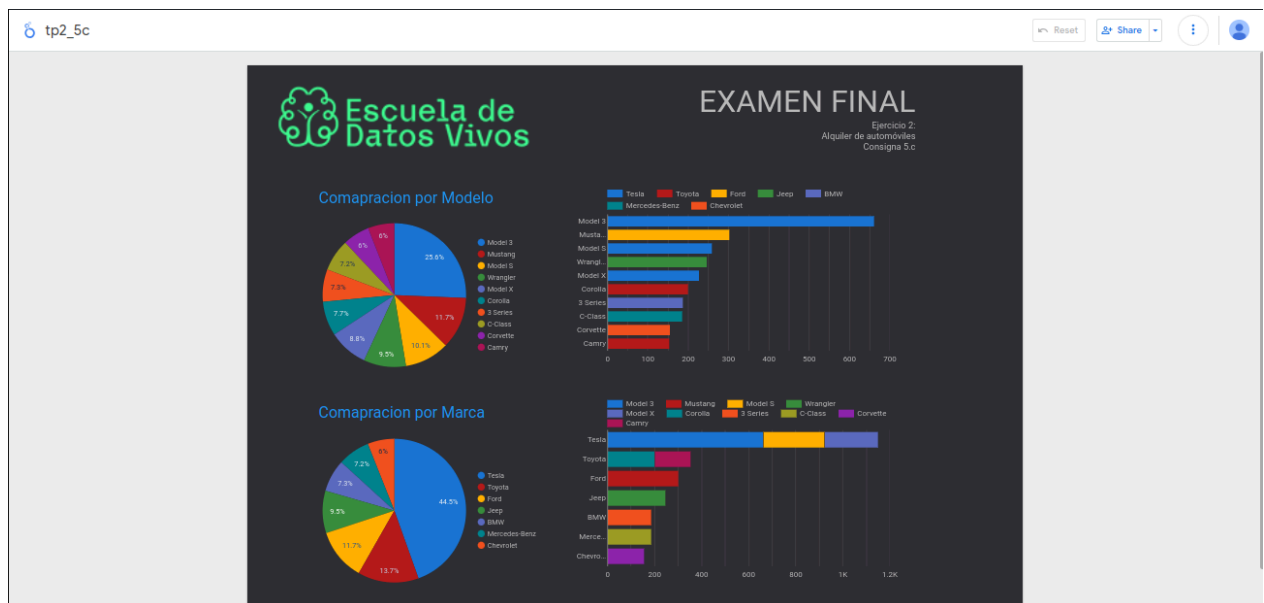
b. los 5 estados con menor cantidad de alquileres (crear visualización)

Link <https://lookerstudio.google.com/reporting/42713e86-8a07-4cab-8220-2b4ba648845b>



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

Link <https://lookerstudio.google.com/reporting/0c2b4151-af7f-4b6c-8ed4-b4c78334d72a>



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

```

SELECT
  year,
  COUNT(make) AS consulta_d
FROM
  car_rental_db.car_rental_analytics
WHERE
  year >= 2010
  AND year <= 2015
GROUP BY year ;

```

Results 1 X

SELECT year COUNT(make) AS Enter a SQL expres

	123 year	123 consulta d
1	2,010	664
2	2,011	952
3	2,012	1,164
4	2,013	1,484
5	2,014	1,812
6	2,015	2,508

- e. las 5 ciudades con más alquileres de vehículos ecológicos (fuelType híbrido o eléctrico)

```

AND year <= 2015,
SELECT
  city AS consulta_e,
  COUNT(city) AS recuento_alquileres_ecologicos
FROM
  car_rental_db.car_rental_analytics
WHERE
  fueltype IN ('hybrid', 'electric')
GROUP BY
  city
ORDER BY
  recuento_alquileres_ecologicos DESC
LIMIT 5;

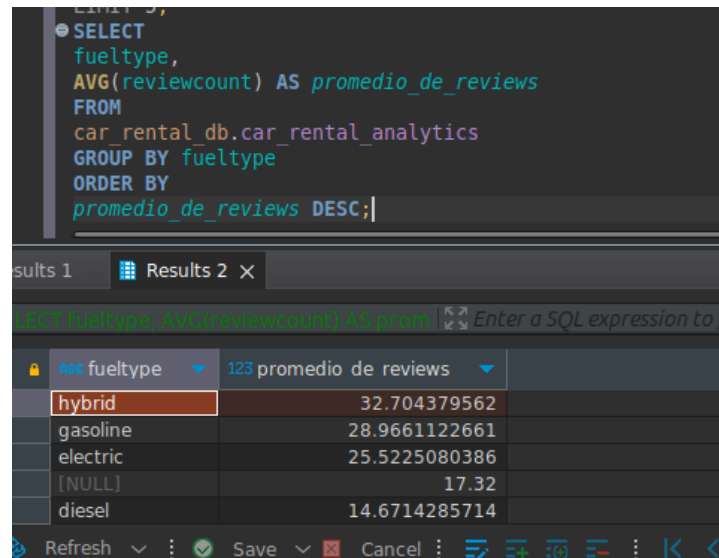
```

Results 1 Results 2 X

SELECT city AS consulta\_e, COUNT(city) AS re Enter a SQL expression to

	123 consulta e	123 recuento alquileres ekologicos
1	San Diego	47
2	Las Vegas	36
3	Portland	21
4	Phoenix	19
5	San Jose	17

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



```

SELECT
fueltype,
AVG(reviewcount) AS promedio_de_reviews
FROM
car_rental_db.car_rental_analytics
GROUP BY fueltype
ORDER BY
promedio_de_reviews DESC;

```

fueltype	promedio de reviews
hybrid	32.704379562
gasoline	28.9661122661
electric	25.5225080386
[NULL]	17.32
diesel	14.6714285714

6. Elabore sus conclusiones y recomendaciones sobre este proyecto.

1 - Una de las tablas que ingresamos al sistema no la usamos. Sería bueno usarla o descartarla. Tampoco tiene sentido usar dos DAGs.

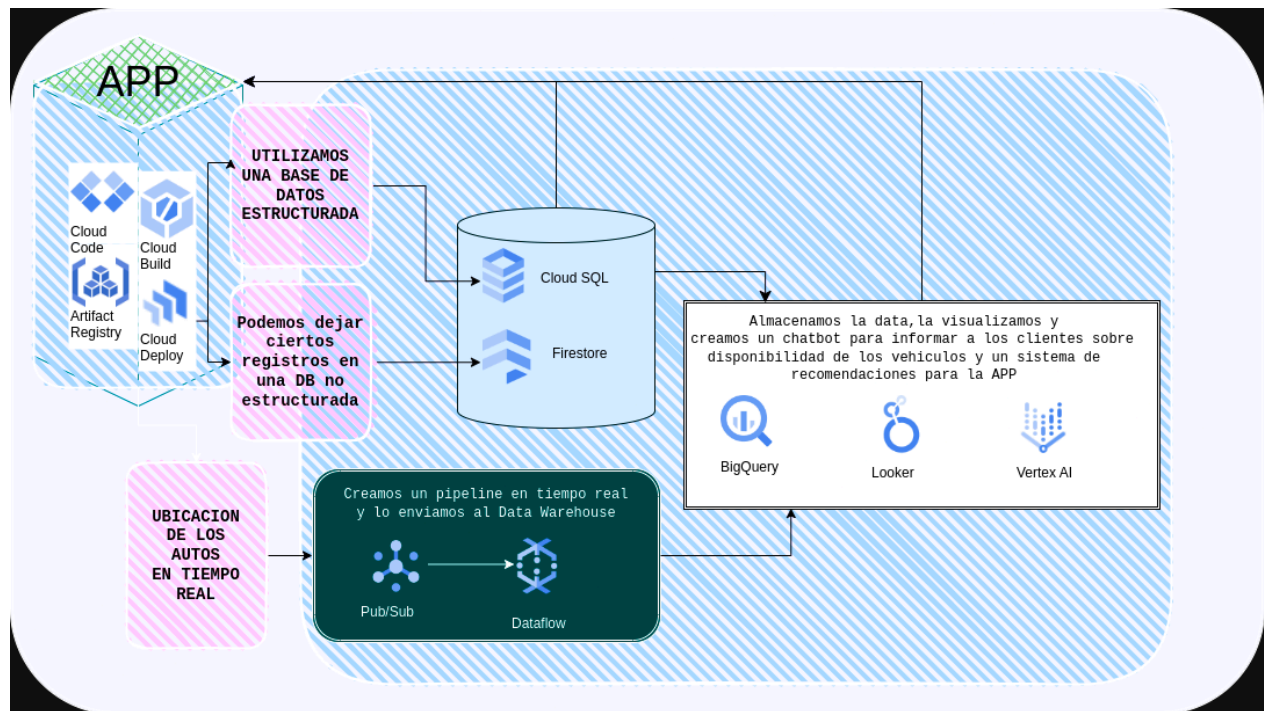
2 - Se puede pasar de un sistema ETL a un ELT si digitalizamos el proceso de alquileres con una APP.

La APP con una ingesta de datos determinada por opciones según su OTLP (Procesamiento Transaccional en Línea), nos puede permitir registrar una data limpia a nuestro OLAP (Procesamiento Analítico en Línea) ahorrando el proceso de transformación.

3 - Tener una gestión de la ubicación de los automóviles en tiempo real nos permite gestionar mejor la devolución de los vehículos en diferentes sucursales, controlar la cantidad de kilómetros realizados y enviar auxilio en caso de accidentes de un modo más ágil.

4 - Con la información recolectada y si estamos utilizando los servicios de GCP, podemos utilizar la herramienta Vertex AI para generar de modo sencillo un chatbot que les permita a los clientes interactuar en todo momento con la información necesaria para su alquiler. A su vez, Vertex AI también tiene disponible un servicio de recomendaciones que puede incrementar la cantidad de alquileres y mejorar el servicio.

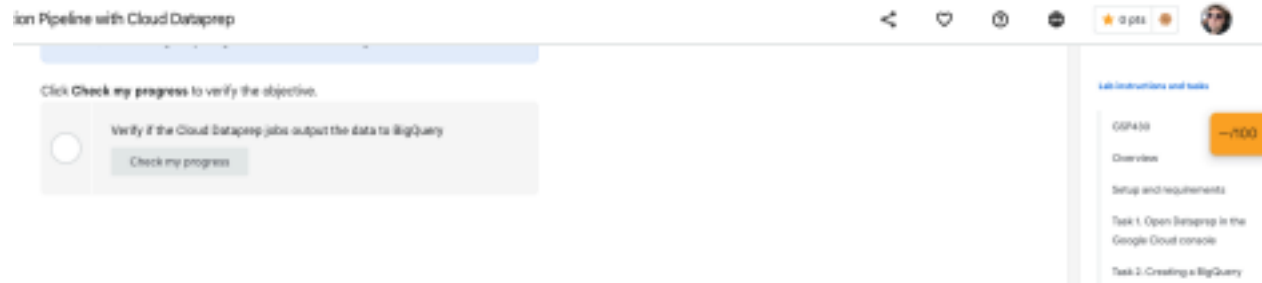
7. Proponer una arquitectura alternativa para este proceso ya sea con herramientas on premise o cloud (Si aplica)



# Google Skills Boost - Examen Final

## LAB:

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



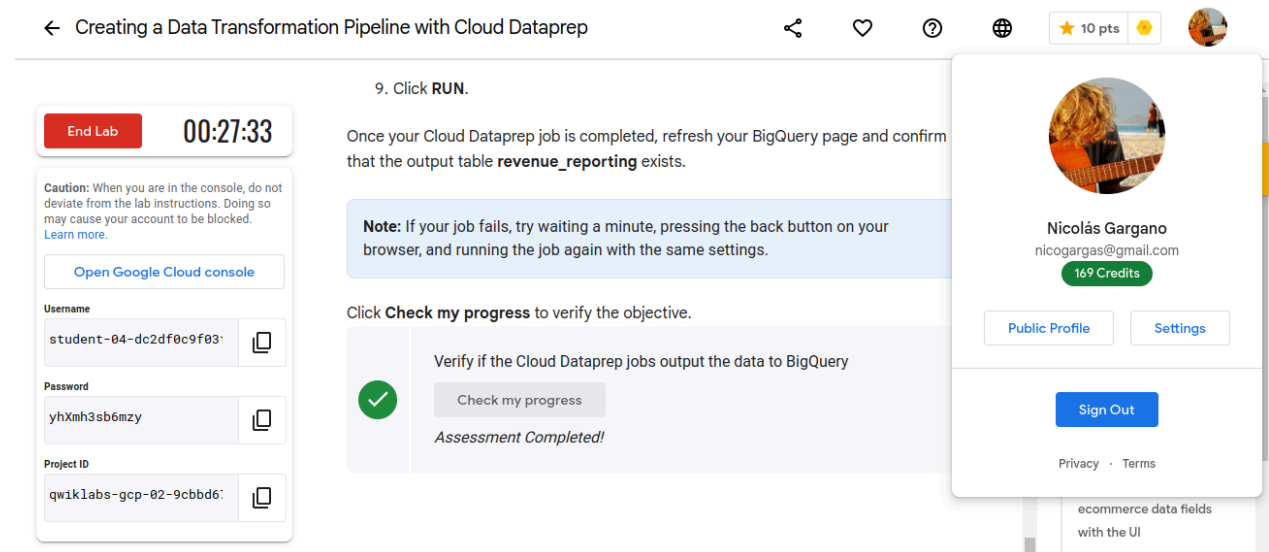
Título: Creating a Data Transformation Pipeline with Cloud Dataprep

Schedule: 1 hour 15 minutes

Cost: 5 Credits

Link:

[https://www.cloudskillsboost.google/focuses/4415?catalog\\_rank=%7B%22rank%22%3A1%2C%22num\\_filters%22%3A0%2C%22has\\_search%22%3Atrue%7D&parent=catalog&search\\_id=32278924](https://www.cloudskillsboost.google/focuses/4415?catalog_rank=%7B%22rank%22%3A1%2C%22num_filters%22%3A0%2C%22has_search%22%3Atrue%7D&parent=catalog&search_id=32278924)



*Contestar las siguientes preguntas:*

*1. ¿Para que se utiliza data prep?*

*Data prep es un servicio no-code desarrollado por Trifacta, para explorar, limpiar y preparar tanto data estructurada como no estructurada. A su vez, su interfaz de usuario permite crear una automatización de ese proceso a través de los servicios de GPC para visualizar los resultados en BigQuery.*



## *2. ¿Qué cosas se pueden realizar con DataPrep?*

- a. Conectarse a conjuntos de datos en BigQuery.*
- b. Explorar la calidad de los datos con una interfaz amigable para el usuario.*
- c. Crear un pipeline de transformación de datos de manera sencilla.*
- d. Ejecutar trabajos para enviar esas transformaciones a BigQuery.*

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

*Por Spark y DataProc, porque son las herramientas que funcionan detrás de DataPrep. Pero dependerá del conocimiento de quien lo use, ya que DataPrep es una herramienta más accesible para usuarios sin experiencia en escribir y ejecutar código de python.*

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

*Considero que los casos de uso comunes incluyen la limpieza de datos, la eliminación de duplicados, la aplicación de filtros, la especificación de tipos de datos, el enriquecimiento de datos, la creación de nuevas columnas basadas en cálculos o uniones, y la normalización de valores numéricos. Pero al no conocer tanto la industria no se realmente cual es el uso general de la herramienta. Ni quienes la usan y cuando.*

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

*Se crea un dataset en BigQuery y se conecta BigQuery con Dataprep en el servicio partner de GPC. Así lo hicimos en el ejercicio.*

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

Tanto estructurada como no estructurada. Pero en el ejercicio trabajamos con data estructurada.

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

a. Limpiar la data

- Borrar las columnas vacías (con todos los datos nulos) o sin usar
- Quitar las filas duplicadas

- Aplicar filtros (nosotros usamos el *WHERE* de SQL).

En el caso del ejercicio nos quedamos con los casos que tienen revenue para aquellos que vieron la página

- Especificar el tipo de dato que necesitamos en cada columna

#### *b. Enriquecer la data*

- Crear nuevas columnas que representen mejor los datos en base a las columnas que ya tenemos. En el caso del ejercicio creamos un *ID* único para cada sesión combinando el *visitid* con el *fullvisitid*.

- Y luego modificamos el código 'número de acciones' de los usuarios por su representación, de este modo nuestra visualización final nos dice la acción, y no el id de la acción. En sql haríamos un JOIN con la tabla acciones para crear una nueva columna que indique la acción tomada y no su código de id*
- Podemos realizar promedios, sumas, u otras operaciones matemáticas. En este caso dividimos totalTransactionRevenue por  $10^6$  para devolverle sus valores originales y que sea un número más razonable*

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

*Tiene la opción RUN. A medida que uno va realizando las transformaciones, estas quedan registradas en una "receta".*

*Una vez terminadas todas las transformaciones que queremos hacer, está la opción "run" que nos envía a un sencillo panel de configuración que crea por medio de una interfaz gráfica todo el proceso en DataFlow de Google.*

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

Por lo que pude observar en el skill boost, visualizaciones detalladas con gráficos y tablas que expresan y describen los datos contenidos en cada columna (valores únicos, distribución y calidad de los datos)



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

En principio, cada columna tiene una barra gris indicando la cantidad de datos faltantes. Pero también, como mencione en la respuesta anterior, uno puede indagar sobre la calidad de los datos haciendo click en la columna y observando detenidamente la visualización de los detalles.

Por otro lado, eso ayuda a que el ingeniero de datos tenga una mejor comprensión sobre las transformaciones que está haciendo y comprenda un poco más la lógica del negocio, especialmente, lo que el cliente necesita, o quiere, o lo que muchas veces sucede, lo que el cliente cree que quiere peeeeero...

## Arquitectura:

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

Esta arquitectura debería contemplar las siguiente etapas:

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

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

**Almacenar:** almacenar los datos procesados en BigQuery

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

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

