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:

```
hive> create database car_rental_db;
OK
Time taken: 2.668 seconds
hive>
```

```
CREATE EXTERNAL TABLE car_rental_db.car_rental_analytics(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)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LOCATION '/tables/external/car_rental_db/car_rental_analytics';
```

```
hive> show tables;
OK
car_rental_analytics
Time taken: 0.049 seconds, Fetched: 1 row(s)
hive>
```

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

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

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

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

wget -P "${ruta}" https://dataengineerpublic.blob.core.windows.net/data-engineer/georef-united-states-of-america-state.csv

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

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

```
hadoop@ec27db0d59e9:~/scripts$ hdfs dfs -ls /ingest
Found 2 items
-rw-r--r--   1 hadoop supergroup     533157 2024-06-18 15:27 /ingest/CarRentalData.csv
-rw-r--r--   1 hadoop supergroup    3380726 2024-06-18 15:27 /ingest/georef-united-states-of-america-state.csv
hadoop@ec27db0d59e9:~/scripts$
```

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**

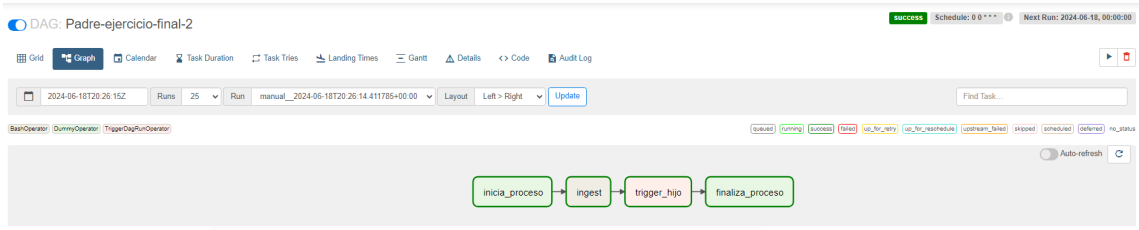
Finalmente insertar en Hive el resultado

```
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)

##leo csv desde HDFS y lo cargo en un dataframe
df = 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/georef-united-states-of-america-state.csv")

##creamos vistas de los df
df.createOrReplaceTempView("CarRentalData")
df2.createOrReplaceTempView("georef")

##filtramos los df
df_RentalData=  spark.sql("select cd.fuelType, CAST(ROUND(CAST(cd.rating as float),0) as int) as rating, CAST(cd.renterTripsTaken as int), CAST(cd.reviewCount as integer), cd.`location.city` as city, gf.`Official
Name State` as state_name, CAST(cd.`owner.id` as int) as owner_id, CAST(cd.`rate.daily` as int) as rate_daily, cd.`vehicle.make`as make, cd.`vehicle.model` as model, CAST(cd.`vehicle.year` as int) as year  from
CarRentalData cd left join georef  gf on cd.`location.state` = gf.`United States Postal Service state abbreviation`")

df_RentalData.createOrReplaceTempView("cast_joined")

df_RentalData =  spark.sql("select LOWER(fuelType), rating, renterTripsTaken, reviewCount, city, state_name, owner_id, rate_daily, make, model, year from cast_joined where rating IS NOT NULL and state_name !=
'Texas' ")

df_RentalData.createOrReplaceTempView("final_table")

##insertamos en Hive
spark.sql("insert into car_rental_db.car_rental_analytics select * from final_table")
```

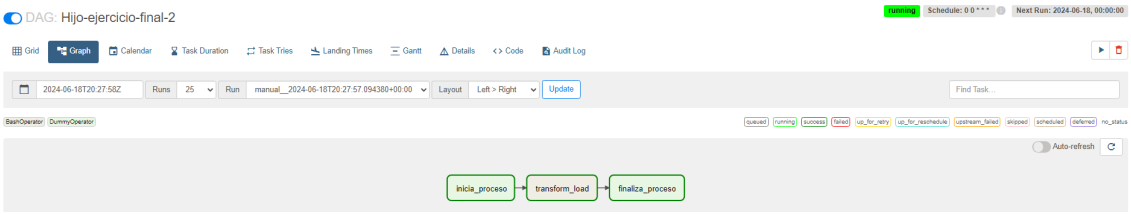
4. Realizar un proceso automático en Airflow que orqueste los pipelines creados en los puntos anteriores. Crear dos tareas:

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

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.operators.trigger_dagrun import TriggerDagRunOperator
from airflow.utils.dates import days_ago

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

with DAG(
    dag_id='Padre-ejercicio-final-2',
    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_2.sh ',
    )

    trigger_hijo = TriggerDagRunOperator(
        task_id="trigger_hijo",
        trigger_dag_id="Hijo-ejercicio-final-2",
        execution_date = '{{ ds }}',
        reset_dag_run = True
    )


    inicia_proceso >> ingest >> trigger_hijo >> finaliza_proceso
```




```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='Hijo-ejercicio-final-2',
    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',

    )

    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_final_2.py ',
    )


    inicia_proceso >> transform_load >> finaliza_proceso
```

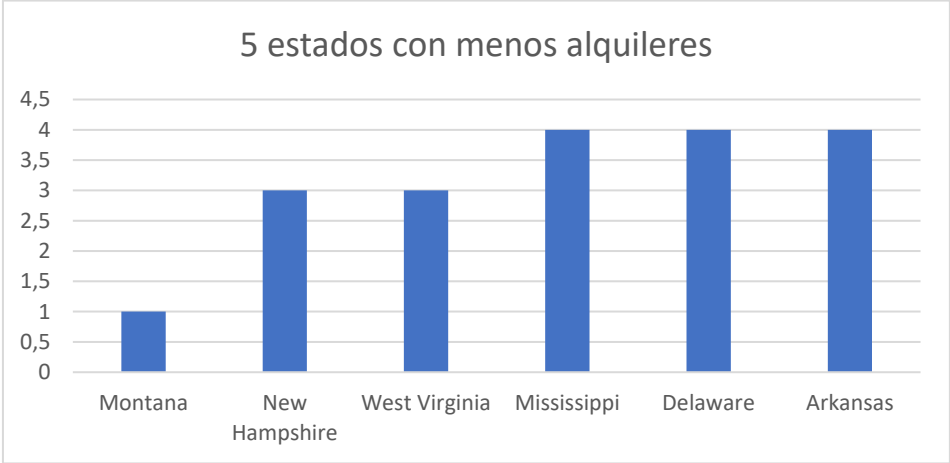
5. Por medio de consultas SQL al data-warehouse, mostrar:

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

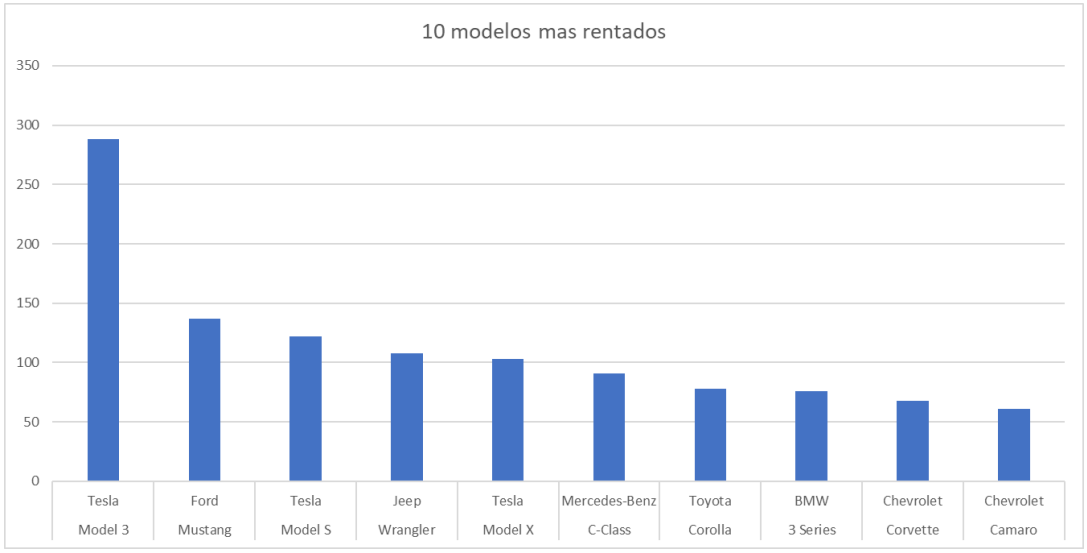
```sql
select count(*) as cantidad_alquileres
from car_rental_analytics cra
where rating >= 4 and (fueltype ='hybrid' or fueltype = 'electric')
```

	123 cantidad_alquileres
1	771

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



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



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

```
select year, count(*) as cantidad_alquileres
from car_rental_analytics cra
where year >= 2010 and year <= 2015
GROUP by year
```

	year	cantidad_alquileres
1	2,010	144
2	2,011	200
3	2,012	225
4	2,013	305
5	2,014	382
6	2,015	532

e. las 5 ciudades con más alquileres de vehículos ecológicos (fuelType hibrido o electrico)

```
select city, count(*) as cantidad_alquileres
from car_rental_analytics cra
where fueltype ='hybrid' or fueltype = 'electric'
GROUP by city
order by cantidad_alquileres desc
limit 5
```

	city	cantidad_alquileres
1	San Diego	44
2	Las Vegas	34
3	Portland	20
4	Phoenix	17
5	San Jose	15

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

```
select fueltype, ROUND(avg(reviewcount),2) as cantidad_reviews
from car_rental_analytics cra
GROUP by fueltype
ORDER by cantidad_reviews desc
```

	fueltype	cantidad_reviews
1	hybrid	34.87
2	gasoline	31.93
3	electric	28.34
4	[NULL]	21.05
5	diesel	17.5

6. Elabore sus conclusiones y recomendaciones sobre este proyecto.

Conclusiones:

-Muestra la preferencia de las personas por lo modelos más modernos.

-La marca más elegida es Tesla, lo que podría marcar la falta de opciones para vehículos eléctricos/hibridos.

- La buena calificación sobre los vehículos eléctricos/híbridos podría marcar la elección hacia opciones más responsables respecto al consumo de combustible en los próximos años.

-La elección de vehículos con combustibles convencionales sigue predominando con preferencia hacia modelos potentes/deportivos.

- Las 5 ciudades con más alquileres de vehículos ecológicos podría verificar que su población tiene una mayor conciencia y demanda por opciones de transportes sostenibles.

- Los 5 estados con menor demanda de alquiler podría significar falta de políticas de turismo, buen servicio de transporte público, etc.

Recomendaciones:

-Evitar siempre datos nulos. Realizar encuesta cerrada que impida la falta de datos.

-Mantener siempre actualizada la flota de autos, podría aumentar la demanda.

-Realizar estudio de mercado para verificar la falta de modelos hibridos/eléctricos o la preferencia sobre una marca en particular.

-Realizar estudio de mercado para verificar la falta de oferta de vehiculos hibridos/eléctricos.

-Investigar la falta de demanda en los estados con menos cantidad de alquileres. Ya sea para aumentar la oferta, corregir políticas, etc.

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