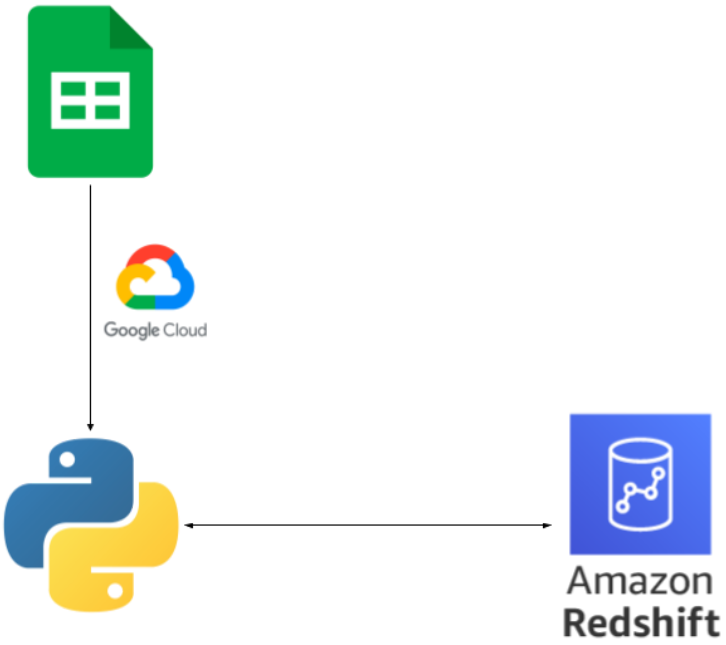
PRUEBA - DATA ENGINEER - GRUPO R5

Néstor Jáuregui

Para la prueba se hizo el montaje de una base de datos usando el motor de SQL Serverless Redshift, se utilizó la API de google para extraer los datos de sheets, y se desarrolló un ETL que corre en un computador local que llama la información utilizando la API de google sheets, transforma los datos y luego los carga a la base de datos. La arquitectura del problema se muestra a continuación:



La arquitectura del problema se planteó de esta forma porque permite que desde el *main* script se unifique tanto el proceso de recopilación de datos como la escritura a la base de datos. Además, la forma en la que se organizaron los directorios y las partes del código permiten agregar otras bases de datos o fuentes de datos de forma tal que el código no se vea afectado en profundidad, y que sea muy fácil de leer. La forma en la que se logró esta ventaja fue usando una estructura de clases que centraliza las bases de datos, al igual de los diferentes *datasets* extraídos de la fuente de datos original.

La fuente de datos se dividió para obtener una estructura de *Base de Datos Relacional* que consiste de diferentes tablas:

- clients: tiene la información de los clientes.
- stores: tiene la información de las tiendas.
- neighborhoods: tiene la información de los barrios.
- purchases: tiene información de las compras.

Además, la forma en la que se dividió la información se hizo de forma tal que se pudiera recrear la información original.

Dentro de la estructura del repositorio se encuentra una carpeta llamada **Python_Pipeline**, en la cual se encuentra toda la estructura con la cual se solucionó el problema de la prueba. El script principal que corre todo el proceso es *main_etl_load_data.py*.

Respuesta a las preguntas:

1. La consulta de SQL utilizada para encontrar la solución fue:

```sql
with base as (
select p.num_documento_cliente,
    p.tipo_documento_cliente,
    p.codigo_tienda,
    p.tipo_tienda,
    s.id_barrio,
    n.nombre_barrio
from purchases p
left join stores s on p.codigo_tienda = s.codigo_tienda
    and p.tipo_tienda = s.tipo_tienda
left join neighborhoods n on s.id_barrio = n.id_barrio
), tienda_cliente as (
select distinct b.codigo_tienda,
    b.num_documento_cliente
from base b
)
select tc.codigo_tienda,
    count(*)
from tienda_cliente tc
group by tc.codigo_tienda
having count(*) = 100
order by tc.codigo_tienda
```

La solución encontrada fue:

codigo_tienda	count
3588	100
4594	100

2. La consulta de SQL utilizada para encontrar la solución fue:

```sql
with base as (
select p.num_documento_cliente,
    p.tipo_documento_cliente,
    p.codigo_tienda,
    p.tipo_tienda,
    s.id_barrio,
    n.nombre_barrio
from purchases p
left join stores s on p.codigo_tienda = s.codigo_tienda
    and p.tipo_tienda = s.tipo_tienda
left join neighborhoods n on s.id_barrio = n.id_barrio
), barrio_tienda as (
select distinct b.id_barrio,
    b.nombre_barrio,
    b.num_documento_cliente
from base b
where b.tipo_tienda = 'Tienda Regional'
)
select bt.id_barrio,
    bt.nombre_barrio,
    count(*)
from barrio_tienda bt
group by bt.id_barrio,
    nombre_barrio
order by count(*) desc
limit 5
```

La solución encontrada fue:

id_barrio	nombre_barrio	count
738000377	Unicentro Cali	354
737998204	Versalles	328
737999419	El Sena	287
737998095	Unidad Residencial Santi...	276
737998838	Calima	250