Veamos un ejemplo de flujo de trabajo, necesitamos obtener algunos datos de un archivo alojado en línea e insertarlos en nuestra base de datos local. También debemos considerar la posibilidad de eliminar filas duplicadas durante la inserción.

Tareas de creación de tablas

Podemos usar PostgresOperator para definir tareas que crean tablas en nuestra base de datos de Postgres.

Crearemos una tabla para facilitar los pasos de limpieza de datos (employees_temp) y otra tabla para almacenar nuestros datos limpios (employees). Crea las dos tareas necesarias para crear ambas tablas ejecutando los siguientes comandos SQL:

1. CREATE TABLE IF NOT EXISTS employees (

```
    "Serial Number" NUMERIC PRIMARY KEY,
    "Company Name" TEXT,
    "Employee Markme" TEXT,
    "Description" TEXT,
    "Leave" INTEGER
);
```

2. DROP TABLE IF EXISTS employees_temp;

```
CREATE TABLE employees_temp (
    "Serial Number" NUMERIC PRIMARY KEY,
    "Company Name" TEXT,
    "Employee Markme" TEXT,
    "Description" TEXT,
    "Leave" INTEGER
);
```

Tarea de recuperación de datos

Aquí recuperamos datos, los guardamos en un archivo en nuestra instancia de Airflow y cargamos los datos de ese archivo en una tabla intermedia donde podemos ejecutar pasos de limpieza de datos.

Obtén los datos desde esta url:

https://raw.githubusercontent.com/apache/airflow/main/docs/apache-airflow/tutorial/pipeline_example.csv

Y guardalos en data_path = "/opt/airflow/dags/files/employees.csv"

Usa el PostgresHook para ejecutar el siguiente comando:

"COPY employees_temp FROM STDIN WITH CSV HEADER DELIMITER AS ',' QUOTE '\"'",

Y cargar el archivo en la tabla creada previamente

Tarea de combinación de datos

Aquí seleccionamos registros completamente únicos de los datos recuperados, luego verificamos si algún empleado ya está en la base de datos (si lo está, actualizamos esos registros con los nuevos datos).Serial Numbers

Incluye una tarea para ejecutar esta Query query = """

```
INSERT INTO employees
SELECT *
FROM (
   SELECT DISTINCT *
   FROM employees_temp
) t
ON CONFLICT ("Serial Number") DO UPDATE
SET
     "Employee Markme" = excluded."Employee Markme",
     "Description" = excluded."Description",
     "Leave" = excluded."Leave";
"""
```

Completando nuestro DAG

Hemos desarrollado nuestras tareas, ahora necesitamos envolverlas en un DAG, lo que nos permite definir cuándo y cómo deben ejecutarse las tareas, e indicar cualquier dependencia que tengan las tareas con otras tareas. Configura el DAG para ejecutarlo todos los días a la medianoche a partir del 1 de enero de 2023.

Y define las siguientes dependencias: La tarea `merge_data()` depende de la tarea `get_data()`, la cual, a su vez, depende tanto de las tareas `create_employees_table` como `create_employees_temp_table`. Además, las tareas `create_employees_table` y `create_employees_temp_table` pueden ejecutarse de forma independiente.