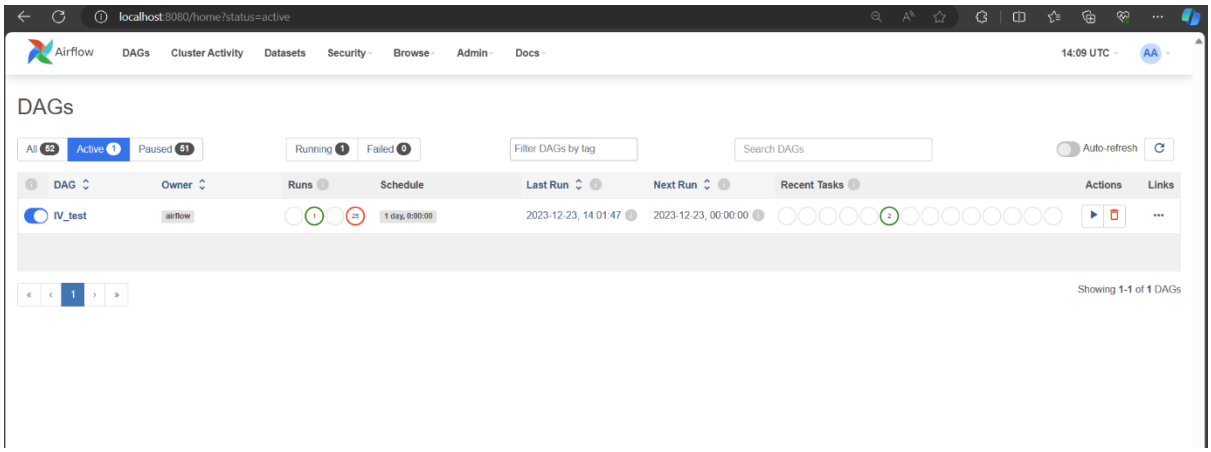
Successfully running the process from Airflow.



Big_query information stored under my own data project

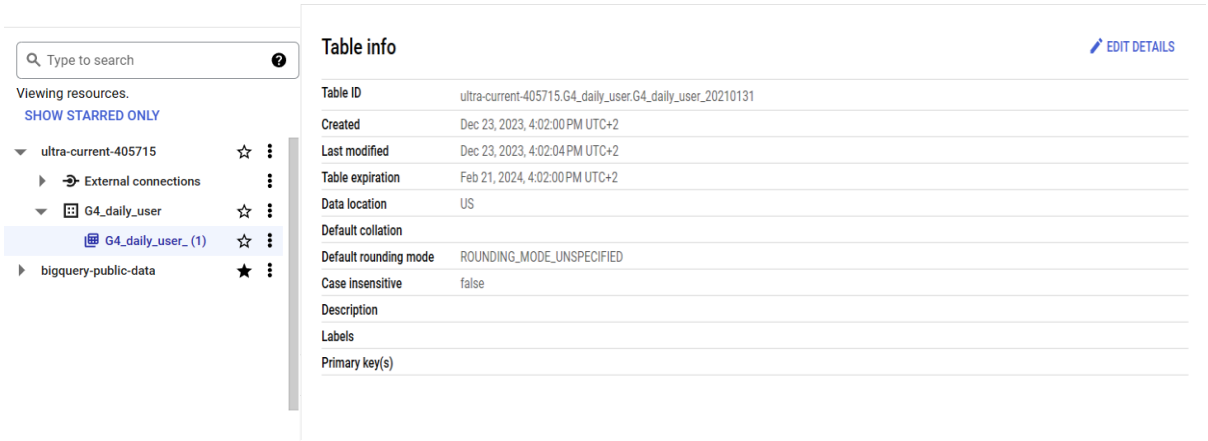


Table information

SCHEMA | DETAILS | PREVIEW | LINEAGE | DATA PROFILE | DATA QUALITY

Row	user_pseudo_id	number_of_even	device	OS	country	revenue
1	4103096.0148768502	176	Edge	Web	Norway	0.0
2	7185511.8469085861	29	Edge	Windows	Canada	0.0
3	8268852.4406584193	93	Edge	Web	South Korea	0.0
4	9029198.5070084673	41	Edge	Web	Netherlands	0.0
5	61927313.4268713170	33	Edge	Web	Thailand	0.0
6	76423298.1901120078	38	Edge	Web	Canada	0.0
7	54824213.8140196606	81	iPad	iOS	United States	0.0
8	82544338.6275608427	33	iPad	Web	Spain	0.0
9	1953654.6738569041	40	Chrome	Windows	United States	0.0
10	2430318.6546688585	33	Chrome	Web	Hong Kong	0.0
11	2717707.1340952423	105	Chrome	Windows	Ireland	0.0
12	2997048.1440031981	36	Chrome	Web	United States	0.0
13	4119699.1554866319	37	Chrome	Windows	Iraq	0.0

DAG Codes :

```
#Import libraries
from google.cloud import bigquery
from airflow.operators.python_operator import PythonOperator
from airflow.contrib.operators.bigquery_check_operator import
BigQueryCheckOperator
import pandas as pd
from google.oauth2 import service_account
from numpy import abs as np_abs
from airflow import DAG
import datetime as dt

#Define arguments
default_args = {
    'retries': 2,
    'retry_delay': dt.timedelta(minutes=1),
    'email_on_retry': False,
    'email_on_failure': False,
}

dag = DAG(
    'IV_test',  # Name that appears in Airflow
    default_args=default_args,
    start_date=dt.datetime(2021, 1, 31),
    schedule_interval=dt.timedelta(days=1),
    catchup=True
)

#Define a function which includes the entire ETL process
def run_etl(ds=None):

    #######  EXCTRACT############
    credentials = service_account.Credentials.from_service_account_file(
        '/opt/airflow/dags/ultra-current-405715-8ef648399261.json')

    projectid = "ultra-current-405715"  # Use your own id!
    sql = """
    SELECT  user_pseudo_id,
            count(*) as number_of_events,
            max(device.mobile_model_name) as device,
            max(device.operating_system) as OS,
            max(geo.country) as country,
            sum(user_ltv.revenue) as revenue
```

```
    FROM `bigquery-public-
data.ga4_obfuscated_sample_ecommerce.events_20210131`
    group by user_pseudo_id
    ORDER BY revenue desc
    """
    df_raw = pd.read_gbq(sql, projectid, credentials=credentials)


    #############    LOAD   ######################
    df_raw.to_gbq('G4_daily_user.G4_daily_user_20210131',
                  project_id='ultra-current-405715', if_exists='replace')


# BigQueryCheckOperator check that the SQL inside returns a single row
# This case basically that the data exists
t1 = BigQueryCheckOperator(
    task_id="check_bigquery",  # Appears in Airflow
    sql="""
    SELECT COUNT(*) FROM `bigquery-public-
data.ga4_obfuscated_sample_ecommerce.events_20210131`
    """,
    use_legacy_sql=False,
    dag=dag,
    # https://www.revisitclass.com/gcp/how-to-configure-google-cloud-bigquery-
connection-in-apache-airflow/
    gcp_conn_id="gcp_bq_con"  # Authentication
)

# Call function run_etl
t2 = PythonOperator(
    task_id="run_etl",  # Appears in Airflow
    python_callable=run_etl,
    dag=dag
)

t1 >> t2  # task 1 needs to be completed before task 2
```