End-to-End ELT Pipeline with Airflow, GCP, BigQuery & Looker Studio

1. Introduction

This project demonstrates the implementation of an automated ELT (Extract, Load, Transform) pipeline using Google Cloud Platform (GCP) services and Apache Airflow. The pipeline processes a global health dataset, loads it into BigQuery, and produces visual insights using Looker Studio.

2. Architecture Overview

Data Flow: GCS → Apache Airflow → BigQuery → Looker Studio

The pipeline extracts a CSV file from a GCS bucket, loads it into BigQuery via Airflow, performs transformations to create country-level tables, and visualizes the data in Looker Studio.

3. Environment Setup & Requirements

The medical research team receives a global health statistics file with disease data by country.

Setting up Airflow Environment

1. Launch a VM instance in GCP: Go to Compute Engine > VM Instances > Create Instance.

2. Name it 'airflow1'. Leave all other configurations at default.

3. Enable HTTPS traffic under Networking.

4. Under Security, allow full access to all Cloud APIs (Airflow needs access to BigQuery).

Installing Airflow on VM

1. SSH into the VM. If blocked, configure the firewall appropriately.

2. Install Python using:

```
sudo apt install python3 python3-pip python3-venv
```

3. Create and activate a virtual environment:

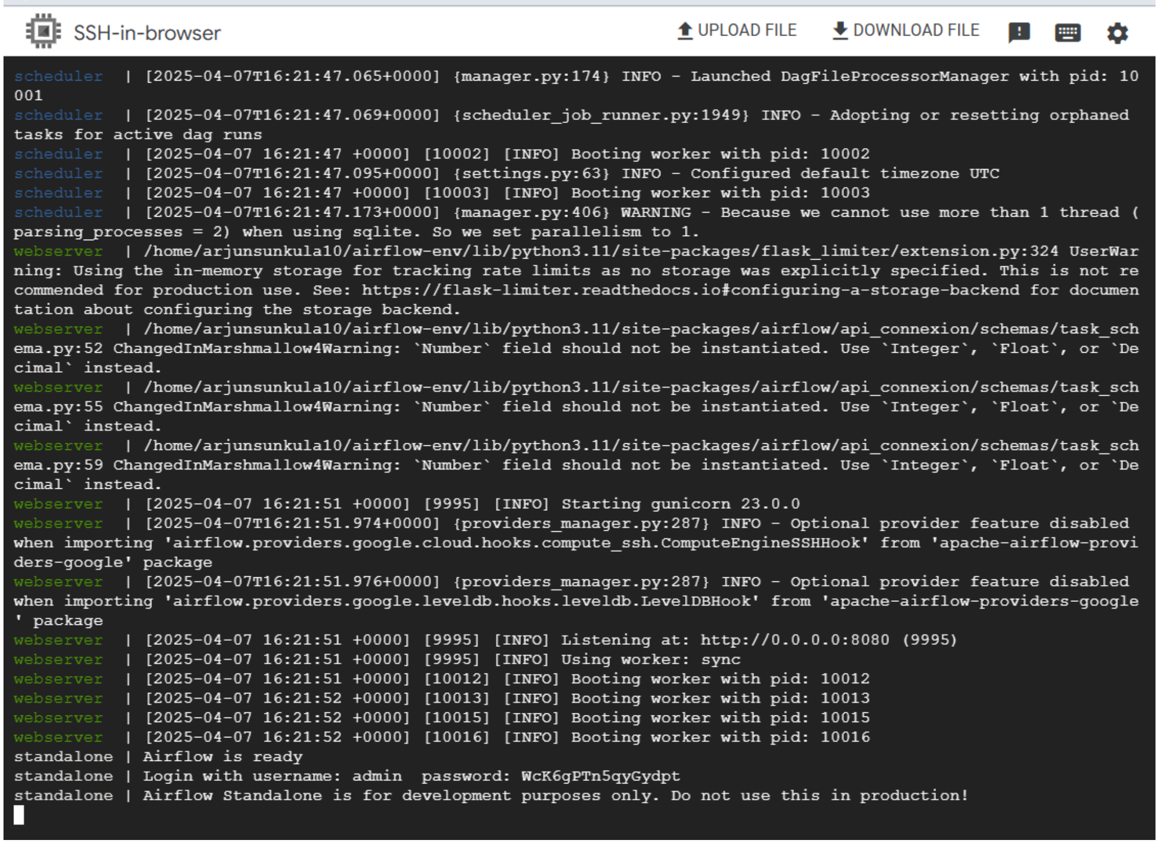
```
python3 -m venv ~/airflow-env

source airflow-env/bin/activate
```

4. Install Airflow with GCP support:

```
pip3 install apache-airflow[gcp]
```

5. Run Airflow in standalone mode:

```
airflow standalone
```



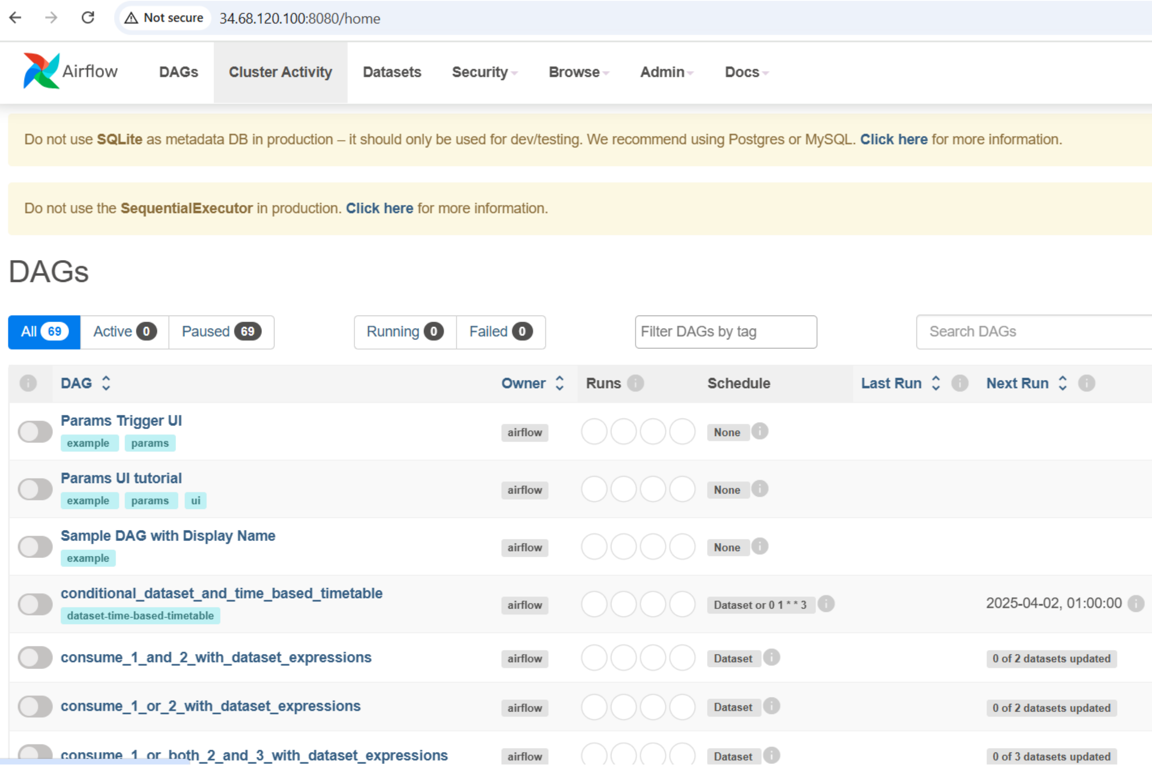
```
scheduler  | [2025-04-07T16:21:47.065+0000] {manager.py:174} INFO - Launched DagFileProcessorManager with pid: 10
001
scheduler  | [2025-04-07T16:21:47.069+0000] {scheduler_job_runner.py:1949} INFO - Adopting or resetting orphaned
tasks for active dag runs
scheduler  | [2025-04-07 16:21:47 +0000] [10002] [INFO] Booting worker with pid: 10002
scheduler  | [2025-04-07T16:21:47.095+0000] {settings.py:63} INFO - Configured default timezone UTC
scheduler  | [2025-04-07 16:21:47 +0000] [10003] [INFO] Booting worker with pid: 10003
scheduler  | [2025-04-07T16:21:47.173+0000] {manager.py:406} WARNING - Because we cannot use more than 1 thread (
parsing_processes = 2) when using sqlite. So we set parallelism to 1.
webserver  | /home/arjunsunkula10/airflow-env/lib/python3.11/site-packages/flask_limiter/extension.py:324 UserWar
ning: Using the in-memory storage for tracking rate limits as no storage was explicitly specified. This is not re
commended for production use. See: https://flask-limiter.readthedocs.io#configuring-a-storage-backend for documen
tation about configuring the storage backend.
webserver  | /home/arjunsunkula10/airflow-env/lib/python3.11/site-packages/airflow/api_connexion/schemas/task_sch
ema.py:52 ChangedInMarshmallow4Warning: `Number` field should not be instantiated. Use `Integer`, `Float`, or `De
cimal` instead.
webserver  | /home/arjunsunkula10/airflow-env/lib/python3.11/site-packages/airflow/api_connexion/schemas/task_sch
ema.py:55 ChangedInMarshmallow4Warning: `Number` field should not be instantiated. Use `Integer`, `Float`, or `De
cimal` instead.
webserver  | /home/arjunsunkula10/airflow-env/lib/python3.11/site-packages/airflow/api_connexion/schemas/task_sch
ema.py:59 ChangedInMarshmallow4Warning: `Number` field should not be instantiated. Use `Integer`, `Float`, or `De
cimal` instead.
webserver  | [2025-04-07 16:21:51 +0000] [9995] [INFO] Starting gunicorn 23.0.0
webserver  | [2025-04-07T16:21:51.974+0000] {providers_manager.py:287} INFO - Optional provider feature disabled
when importing 'airflow.providers.google.cloud.hooks.compute_ssh.ComputeEngineSSHHook' from 'apache-airflow-provi
ders-google' package
webserver  | [2025-04-07T16:21:51.976+0000] {providers_manager.py:287} INFO - Optional provider feature disabled
when importing 'airflow.providers.google.leveldb.hooks.leveldb.LevelDBHook' from 'apache-airflow-providers-google
' package
webserver  | [2025-04-07 16:21:51 +0000] [9995] [INFO] Listening at: http://0.0.0.0:8080 (9995)
webserver  | [2025-04-07 16:21:51 +0000] [9995] [INFO] Using worker: sync
webserver  | [2025-04-07 16:21:51 +0000] [10012] [INFO] Booting worker with pid: 10012
webserver  | [2025-04-07 16:21:52 +0000] [10013] [INFO] Booting worker with pid: 10013
webserver  | [2025-04-07 16:21:52 +0000] [10015] [INFO] Booting worker with pid: 10015
webserver  | [2025-04-07 16:21:52 +0000] [10016] [INFO] Booting worker with pid: 10016
standalone | Airflow is ready
standalone | Login with username: admin  password: WcK6gPTn5qyGydpt
standalone | Airflow Standalone is for development purposes only. Do not use this in production!
```

6. Note the generated username and password for logging into the Airflow UI.

7. To run Airflow in the background:

```
nohup airflow standalone > airflow.log &
```

8. Ensure port 8080 is open in the firewall to access the Airflow UI.



Do not use **SQLite** as metadata DB in production – it should only be used for dev/testing. We recommend using Postgres or MySQL. **Click here** for more information.

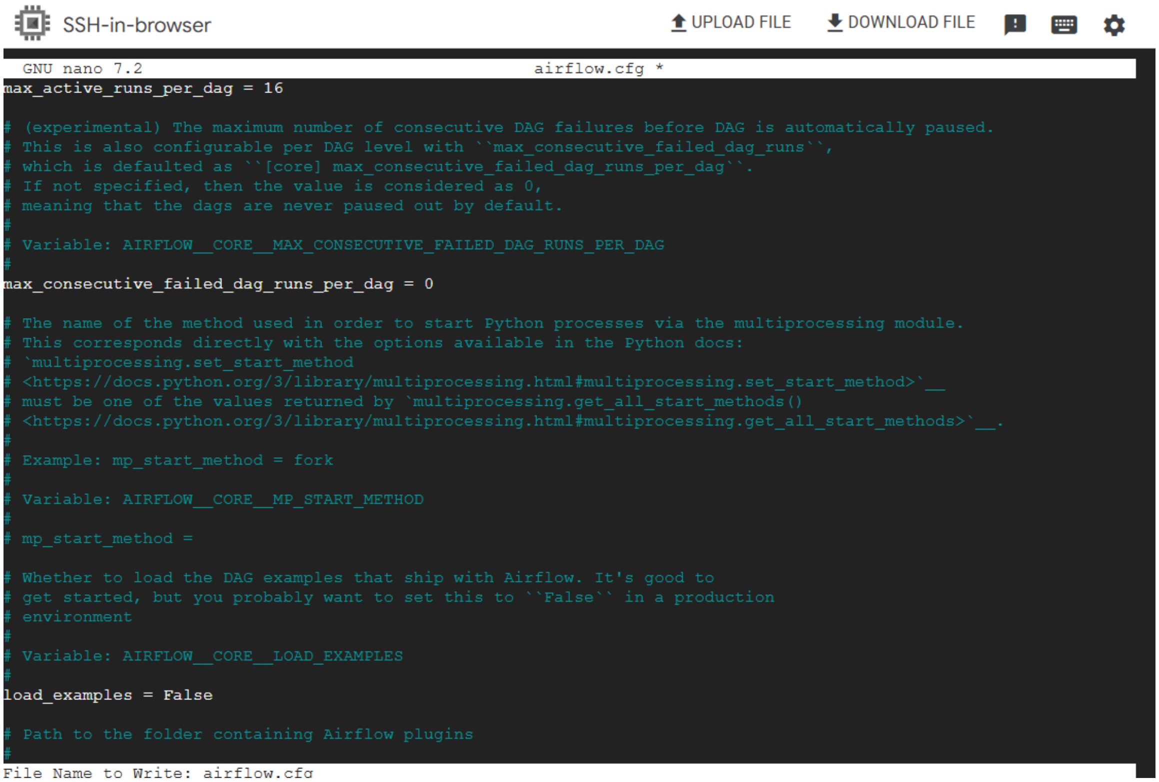
Do not use the **SequentialExecutor** in production. **Click here** for more information.

DAGs

DAG	Owner	Runs	Schedule	Last Run	Next Run
Params Trigger UI example params	airflow		None		
Params UI tutorial example params ui	airflow		None		
Sample DAG with Display Name example	airflow		None		
conditional_dataset_and_time_based_timetable dataset-time-based-timetable	airflow		Dataset or 0 1 * * 3		2025-04-02, 01:00:00
consume_1_and_2_with_dataset_expressions	airflow		Dataset		0 of 2 datasets updated
consume_1_or_2_with_dataset_expressions	airflow		Dataset		0 of 2 datasets updated
consume_1_or_both_2_and_3_with_dataset_expressions	airflow		Dataset		0 of 3 datasets updated

Optimizing Airflow Interface

Disable example DAGs in the Airflow config:



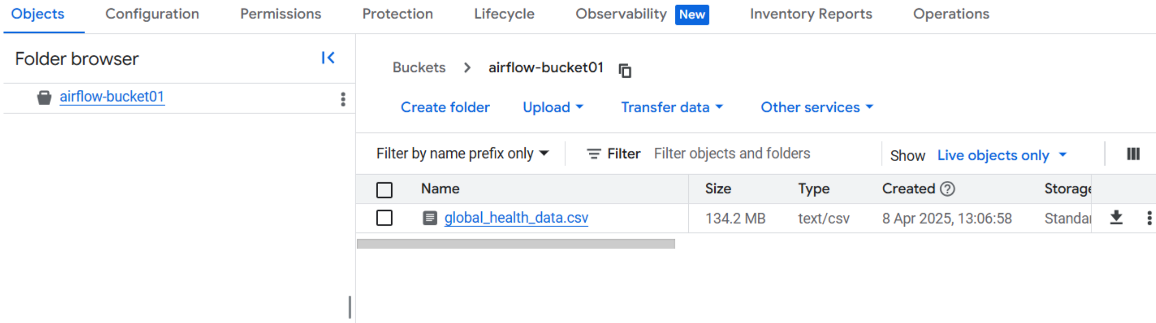
```
  GNU nano 7.2                                        airflow.cfg *
max_active_runs_per_dag = 16

# (experimental) The maximum number of consecutive DAG failures before DAG is automatically paused.
# This is also configurable per DAG level with ``max_consecutive_failed_dag_runs``,
# which is defaulted as ``[core] max_consecutive_failed_dag_runs_per_dag``.
# If not specified, then the value is considered as 0,
# meaning that the dags are never paused out by default.
#
# Variable: AIRFLOW__CORE__MAX_CONSECUTIVE_FAILED_DAG_RUNS_PER_DAG
#
max_consecutive_failed_dag_runs_per_dag = 0

# The name of the method used in order to start Python processes via the multiprocessing module.
# This corresponds directly with the options available in the Python docs:
# `multiprocessing.set_start_method
# <https://docs.python.org/3/library/multiprocessing.html#multiprocessing.set_start_method>`__
# must be one of the values returned by `multiprocessing.get_all_start_methods()
# <https://docs.python.org/3/library/multiprocessing.html#multiprocessing.get_all_start_methods>`__.
#
# Example: mp_start_method = fork
#
# Variable: AIRFLOW__CORE__MP_START_METHOD
#
# mp_start_method =

# Whether to load the DAG examples that ship with Airflow. It's good to
# get started, but you probably want to set this to ``False`` in a production
# environment
#
# Variable: AIRFLOW__CORE__LOAD_EXAMPLES
#
load_examples = False

# Path to the folder containing Airflow plugins
#
File Name to Write: airflow.cfg
```

Set `load_examples = False` in airflow.cfg

Note the `dags_folder` path (e.g., /home/arjunsunkula10/airflow/dags)

4. GCS Bucket Setup

Create a bucket named 'airflow-bucket01' and upload 'global_health_data.csv'.



☐	Name	Size	Type	Created	Storag
☐	global_health_data.csv	134.2 MB	text/csv	8 Apr 2025, 13:06:58	Standa

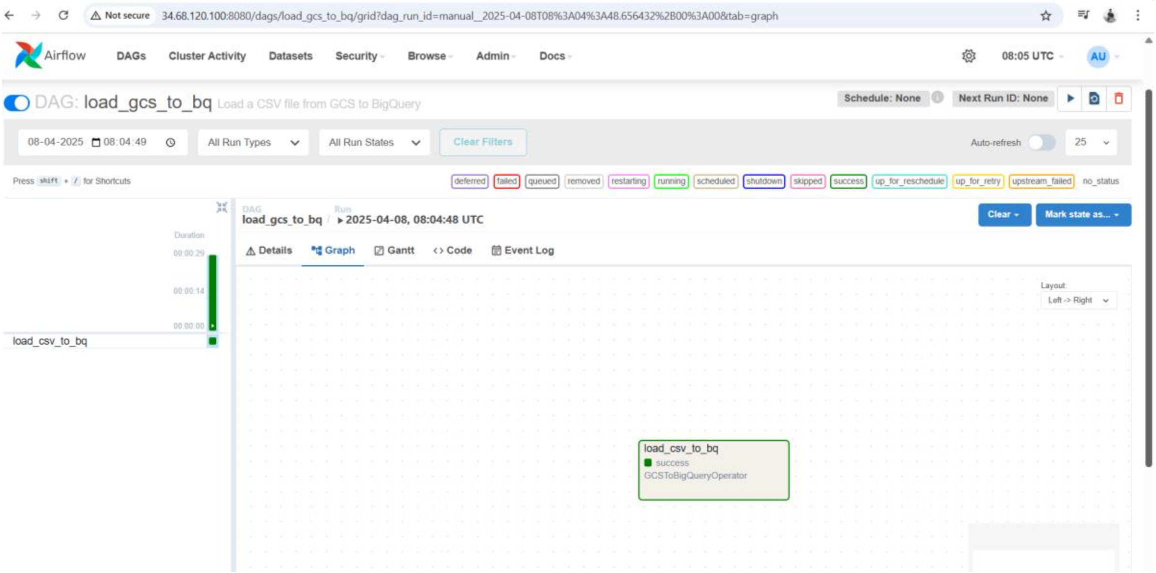
5. BigQuery Loading

1. In BigQuery, create a dataset in your project.

2. Load the CSV file from GCS into a table at the path:

lateral-rider-456109-s6.initial_dataset.global_data

3. Upload the DAG to the Airflow VM to automate loading.



Not secure 34.68.120.100:8080/dags/load_gcs_to_bq/grid?dag_run_id=manual__2025-04-08T08%3A04%3A48.656432%2B00%3A00&tab=graph

DAG: load_gcs_to_bq Load a CSV file from GCS to BigQuery

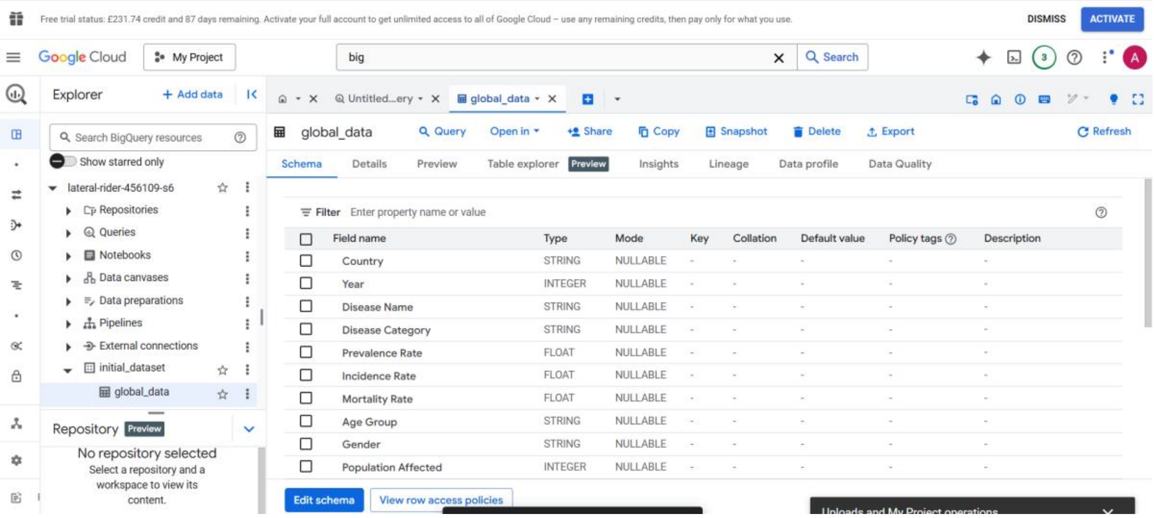
Schedule: None

Next Run ID: None

DAG load_gcs_to_bq / Run 2025-04-08, 08:04:48 UTC

load_csv_to_bq
success
GCSToBigQueryOperator

4. Confirm data is visible in BigQuery.



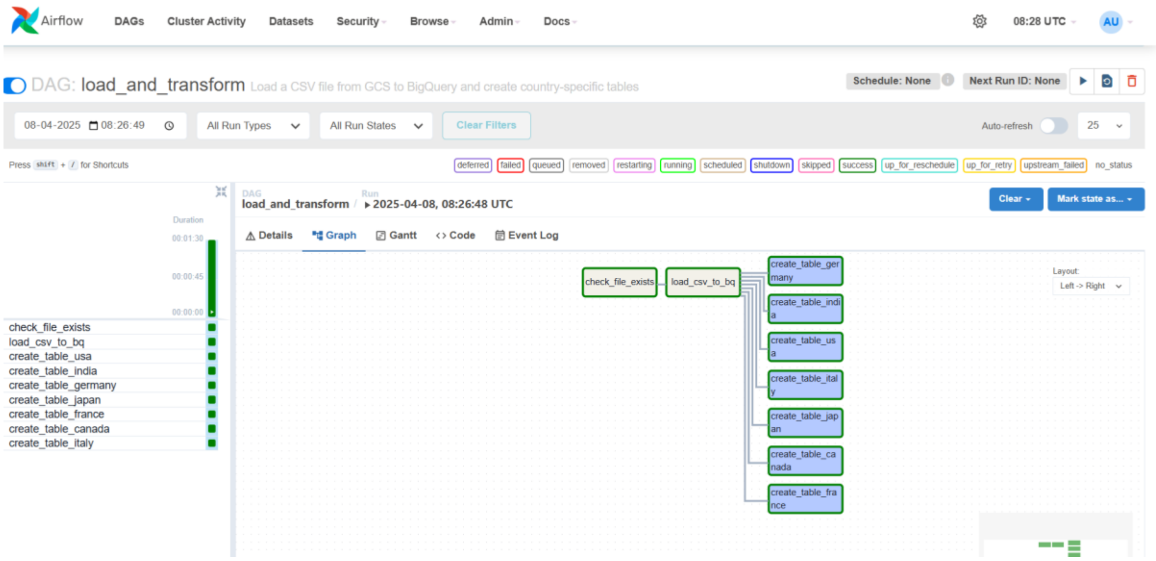
Free trial status: £231.74 credit and 87 days remaining. Activate your full account to get unlimited access to all of Google Cloud – use any remaining credits, then pay only for what you use.

Field name	Type	Mode	Key	Collation	Default value	Policy tags	Description
Country	STRING	NULLABLE	-	-	-	-	-
Year	INTEGER	NULLABLE	-	-	-	-	-
Disease Name	STRING	NULLABLE	-	-	-	-	-
Disease Category	STRING	NULLABLE	-	-	-	-	-
Prevalence Rate	FLOAT	NULLABLE	-	-	-	-	-
Incidence Rate	FLOAT	NULLABLE	-	-	-	-	-
Mortality Rate	FLOAT	NULLABLE	-	-	-	-	-
Age Group	STRING	NULLABLE	-	-	-	-	-
Gender	STRING	NULLABLE	-	-	-	-	-
Population Affected	INTEGER	NULLABLE	-	-	-	-	-

6. File Detection and Data Transformation

Implement a sensor DAG to verify file availability in the bucket before proceeding with the load.

Upon confirmation, the data is processed to generate country-specific tables.



DAG: load_and_transform Load a CSV file from GCS to BigQuery and create country-specific tables

Schedule: None

Next Run ID: None

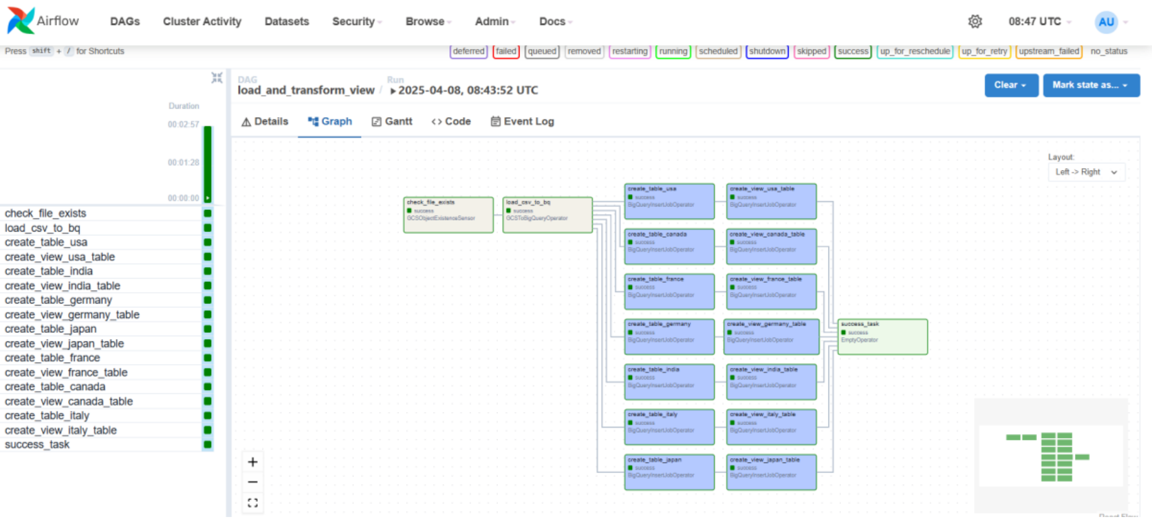
08-04-2025 08:26:49

DAG load_and_transform / Run 2025-04-08, 08:26:48 UTC

7. Creating Views for Optimization

Build views on top of transformed tables to support analytical queries.

These views reduce query size and enhance performance by focusing on filtered fields.



DAG load_and_transform_view / Run ▸2025-04-08, 08:43:52 UTC

Layout: Left -> Right

check_file_exists 🟩
load_csv_to_bq 🟩
create_table_usa 🟩
create_view_usa_table 🟩
create_table_india 🟩
create_view_india_table 🟩
create_table_germany 🟩
create_view_germany_table 🟩
create_table_japan 🟩
create_view_japan_table 🟩
create_table_france 🟩
create_view_france_table 🟩
create_table_canada 🟩
create_view_canada_table 🟩
create_table_italy 🟩
create_view_italy_table 🟩
success_task 🟩

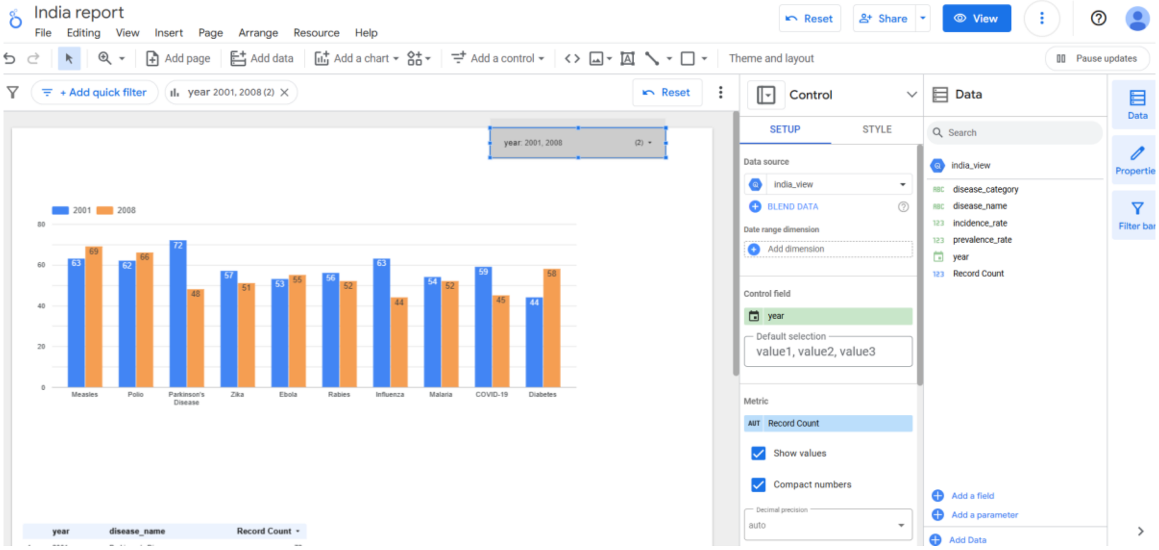
check_file_exists 🟩 success GCSObjectExistenceSensor
load_csv_to_bq 🟩 success GCSToBigQueryOperator
create_table_usa 🟩 success BigQueryInsertJobOperator
create_view_usa_table 🟩 success BigQueryInsertJobOperator
create_table_canada 🟩 success BigQueryInsertJobOperator
create_view_canada_table 🟩 success BigQueryInsertJobOperator
create_table_france 🟩 success BigQueryInsertJobOperator
create_view_france_table 🟩 success BigQueryInsertJobOperator
create_table_germany 🟩 success BigQueryInsertJobOperator
create_view_germany_table 🟩 success BigQueryInsertJobOperator
create_table_india 🟩 success BigQueryInsertJobOperator
create_view_india_table 🟩 success BigQueryInsertJobOperator
create_table_italy 🟩 success BigQueryInsertJobOperator
create_view_italy_table 🟩 success BigQueryInsertJobOperator
create_table_japan 🟩 success BigQueryInsertJobOperator
create_view_japan_table 🟩 success BigQueryInsertJobOperator
success_task 🟩 success EmptyOperator

8. Visualization and Reporting

Connect Looker Studio to BigQuery and create reports based on view tables.

Sample visualizations include disease trends by year and country.

Automated email alerts can be configured to share report summaries.



year	disease_name	Record Count ▾

Data source: india_view

BLEND DATA

Date range dimension: Add dimension

Control field: year

Default selection: value1, value2, value3

Metric: AUT Record Count

☑ Show values

☑ Compact numbers

Decimal precision: auto

india_view
- ABC disease_category
- ABC disease_name
- 123 incidence_rate
- 123 prevalence_rate
- year
- 123 Record Count

Add a field

Add a parameter

Add Data

9. Conclusion

The pipeline provides a complete ELT solution from data ingestion to visualization. It supports reliable data validation, scalable storage, and cost-efficient querying.