Build ELT Data Pipeline to Process 1 million Records with GCP & Airflow | Data Engineering Project

(I created this document to track my steps while implementing this Mini project)

Tools used:

1. BigQuery
2. Airflow
3. GCP

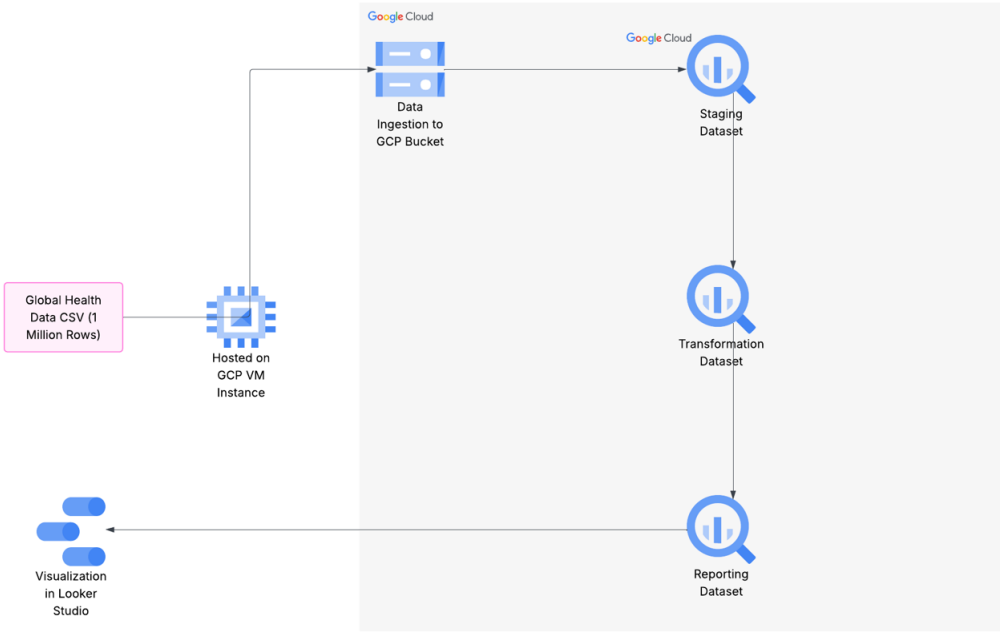
Objective

Break out the CSV file into chunks of data file based on the country

Steps:

- Extract and store raw data in GCS in CSV format.
- Load raw data into a staging table in BigQuery.
- Transform data into country-specific tables and reporting views.
- Use Apache Airflow to orchestrate the pipeline.
- Generate clean and structured datasets for analysis.
- Generate a simple report in Looker Studio

Architecture



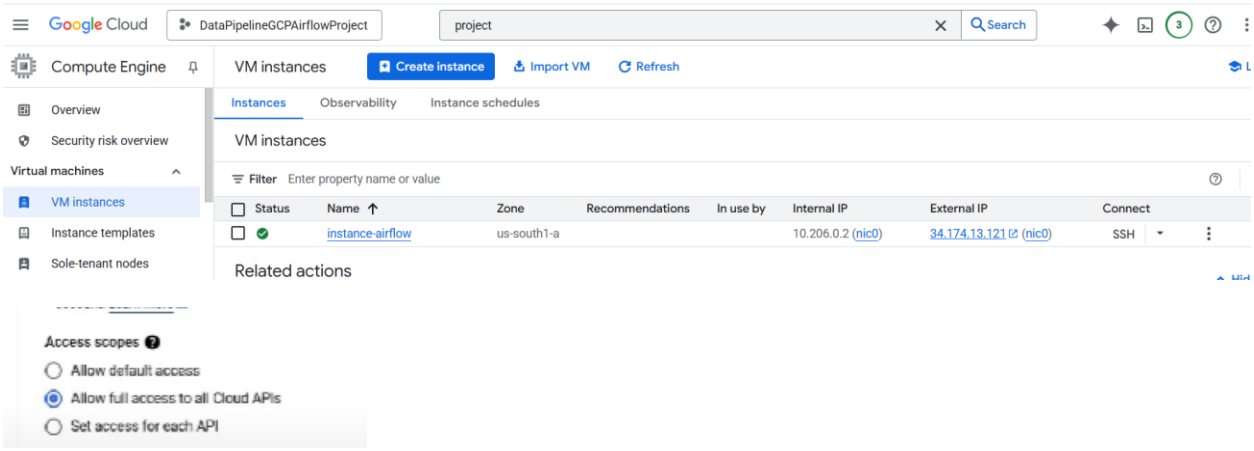
Data set has 1M rows with all countries (source Kaggle)

Google compute engine – is basically a scalable VM based on the workload, it's a Infrastructure As a Service (IAAS) by Google

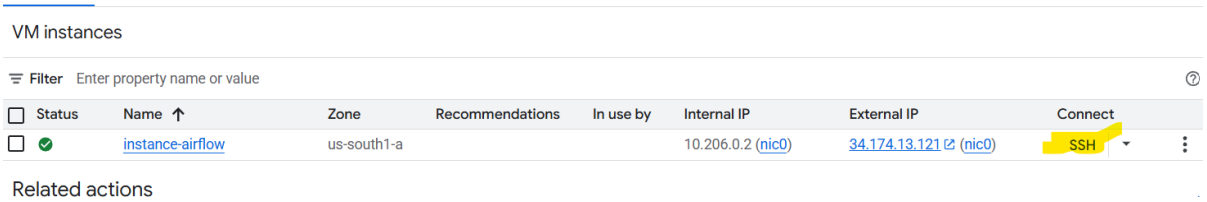
Steps:

1. Set up a Virtual Machine using Google Cloud compute – default settings and allow all access to Cloud APIs so that read and write both permissions are there

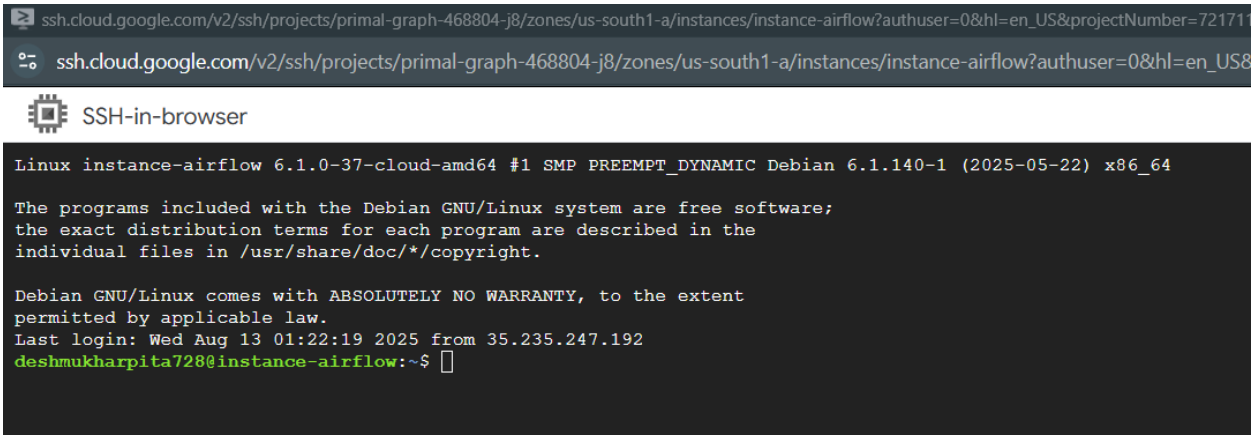
Note: I have used a VM because it easier to set up everything on the VM and get started right away



2. Check if Firewall is set up in SSH (it is setup by default) the click on the SSH button



This should open – access to the VM



3. Set up Airflow on the VM
   Run the below commands

- sudo apt update
  - sudo apt install python3 python3-pip python3-venv
- python3 –m venv ~/airflow-env **(create virtual environment)**
- source airflow-env/bin/activate **(activate the virtual environment)**
- pip3 install apache-airflow

<u>Starting Airflow</u>

Command used to start airflow – **airflow standalone**

NOTE ***- this command gives a username and a password

If it does not give the username and password within 5 mins look for a msg like this -

standalone | Starting Airflow Standalone standalone | Password for the admin user has been previously generated in /home/user/airflow/simple_auth_manager_passwords.json.generated. Not echoing it here.

***go to the location and cat the filename mentioned and copy paste the username password

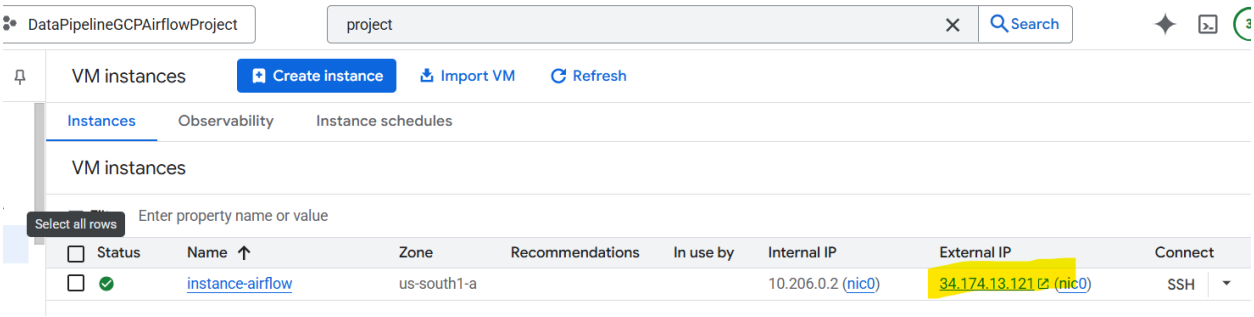
{"admin": "4MFVYndk9UfqxMQk"}

- nohup airflow standalone > airflow.log

  Explantion from ChatGPT -

- **nohup** → *No Hang Up* — runs the command so it ignores the "hangup" signal if you close your terminal or log out.
- **`airflow standalone`** → starts Apache Airflow in standalone mode (scheduler, webserver, DB init, all in one process).
- **> airflow.log** → redirects all output (stdout) into a file named `airflow.log`.

4. Use the username and password to go to the Airflow webpage using the external IP from the VM



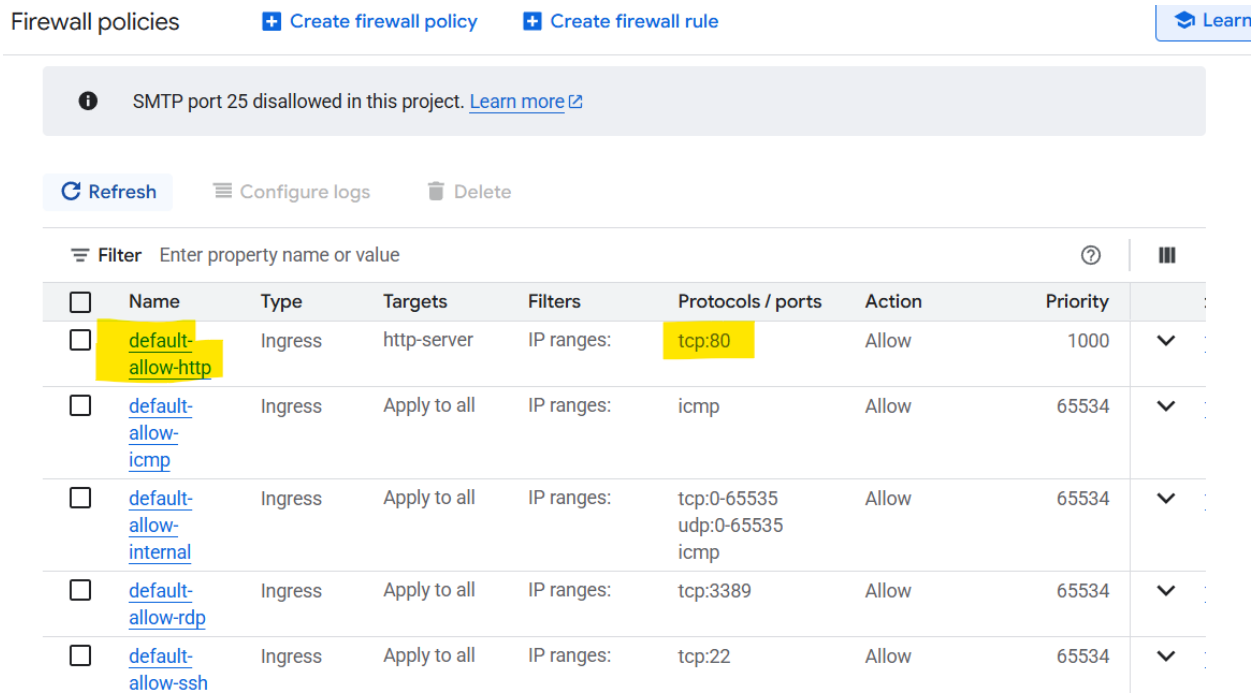
Need to type http://ipaddress:8080

***8080 is the default airflow port

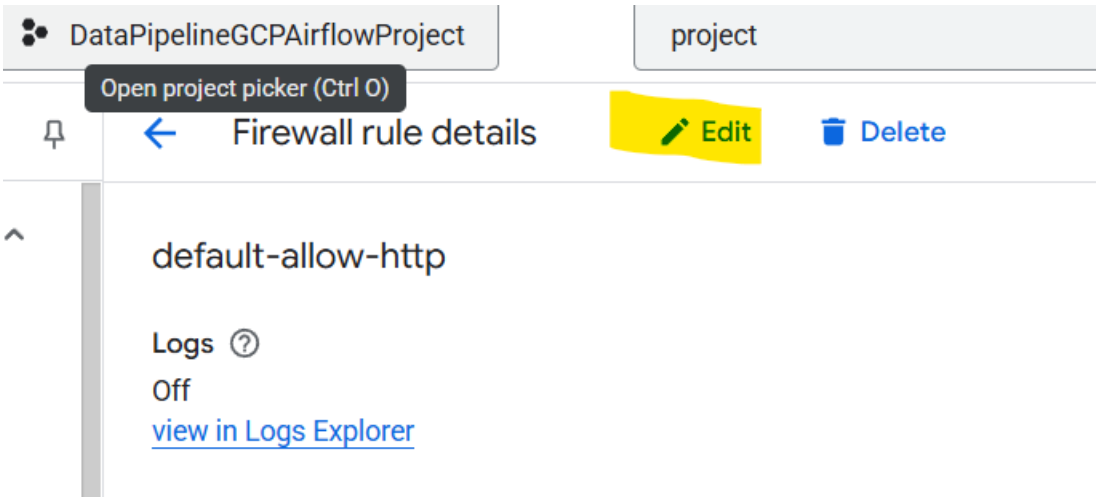
If the browser is not allowing access then need to check network setting and allow access to 8080

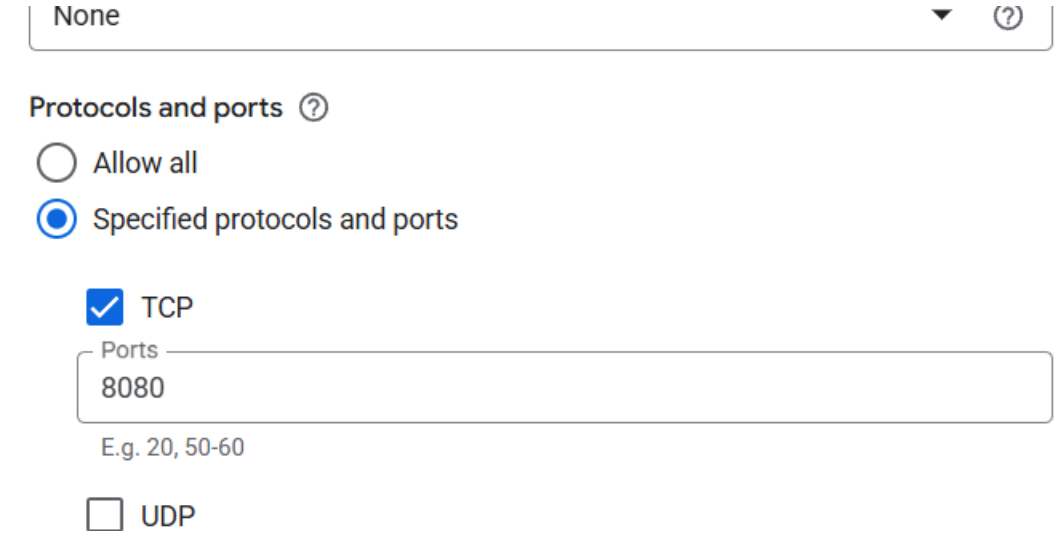
Go to Network > Firewalls

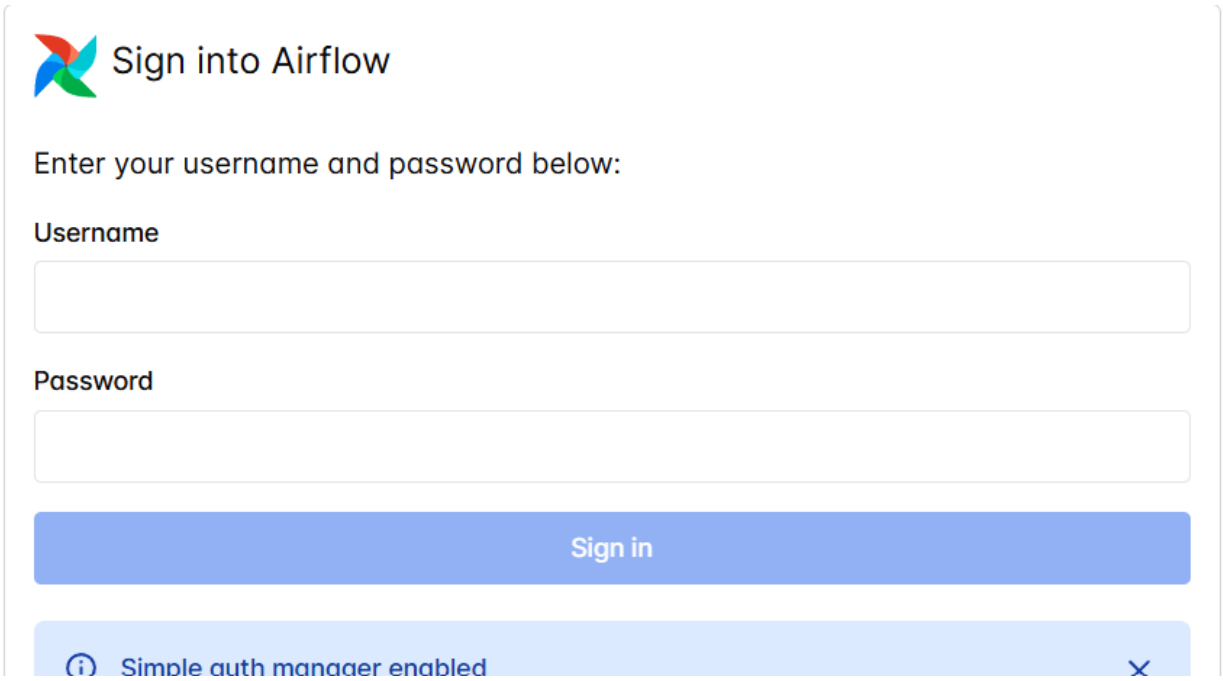
Need to give access to 8080 but right now it is only allowing 80



Hit edit and change it to 8080, hit save. Refresh the Airflow webpage





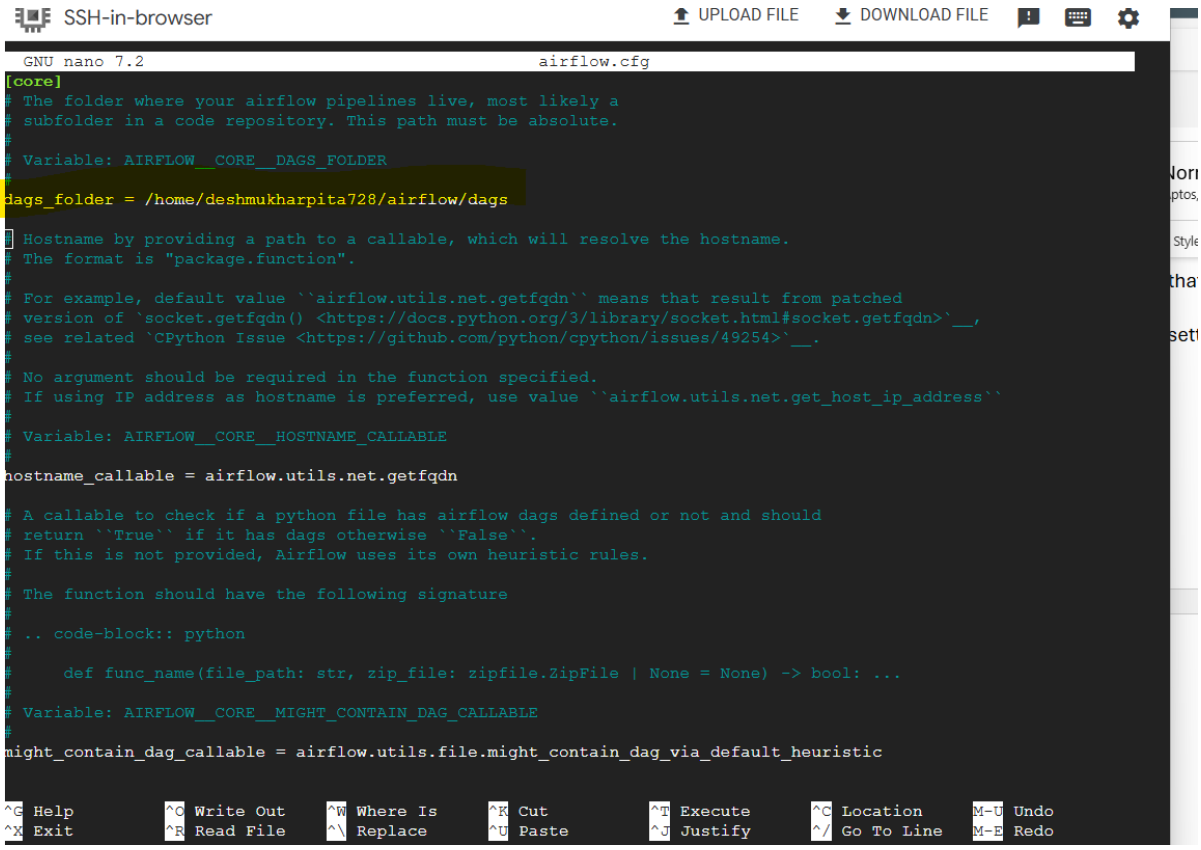


5. Once you close your SSH terminal, the airflow web page will crash that's why the nohup command helps
   - Open a new connection on the current SSH terminal – top right setting tab
   - Close the previous one after typing the nohup command
6. Open the airflow.cfg file
   In the new SSH terminal
   -cd airflow

-nano airflow.cfg

The location of dags folder can be spotted here

dags_folder = /home/deshmukharpita728/airflow/dags

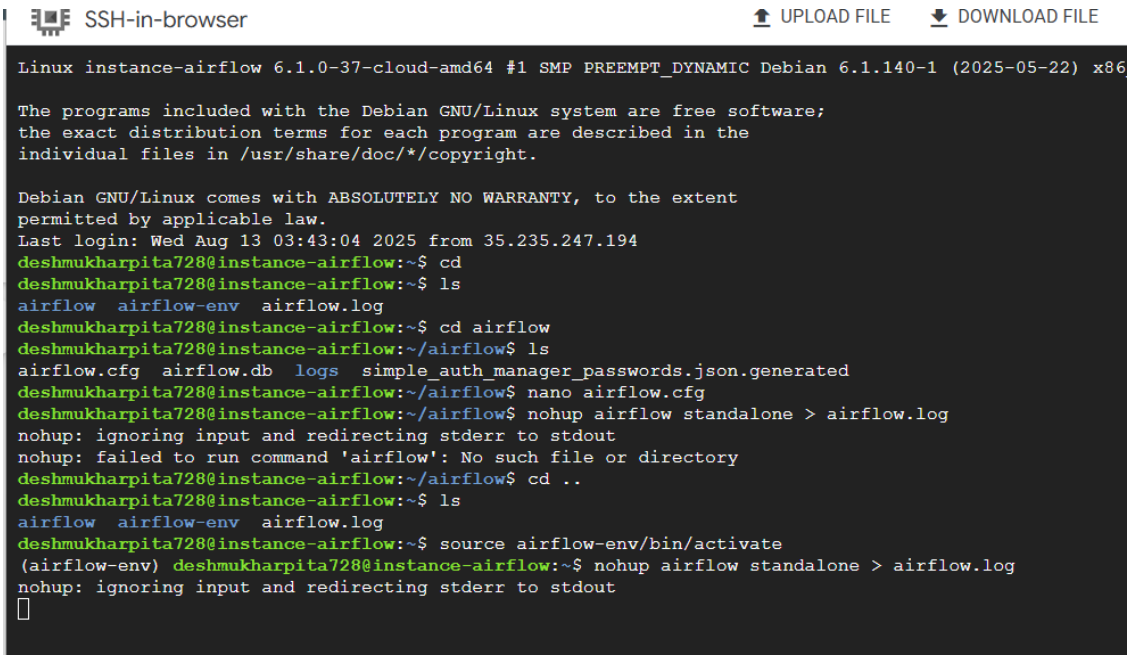


**NOTE: if you want to hide the example DAGs on the Airflow web page

Set the load_examples = False in the config file (initially it was True)

```
#
# Variable: AIRFLOW__CORE__LOAD_EXAMPLES
#
load_examples = False

# Path to the folder containing Airflow plugins
#
# Variable: AIRFLOW__CORE__PLUGINS_FOLDER
#
plugins_folder = /home/deshmukharpita728/airflow/plugins

# Should tasks be executed via forking of the parent process
```

Save and Exit this cfg file. Now go back and write the nohup command again



7. Reload Airflow and all the example DAGs should go away
8. If the UI is not visible, try this

Activate your environment

source ~/airflow-env/bin/activate

Set Airflow to listen on all interfaces

export AIRFLOW__WEBSERVER__WEB_SERVER_HOST=0.0.0.0

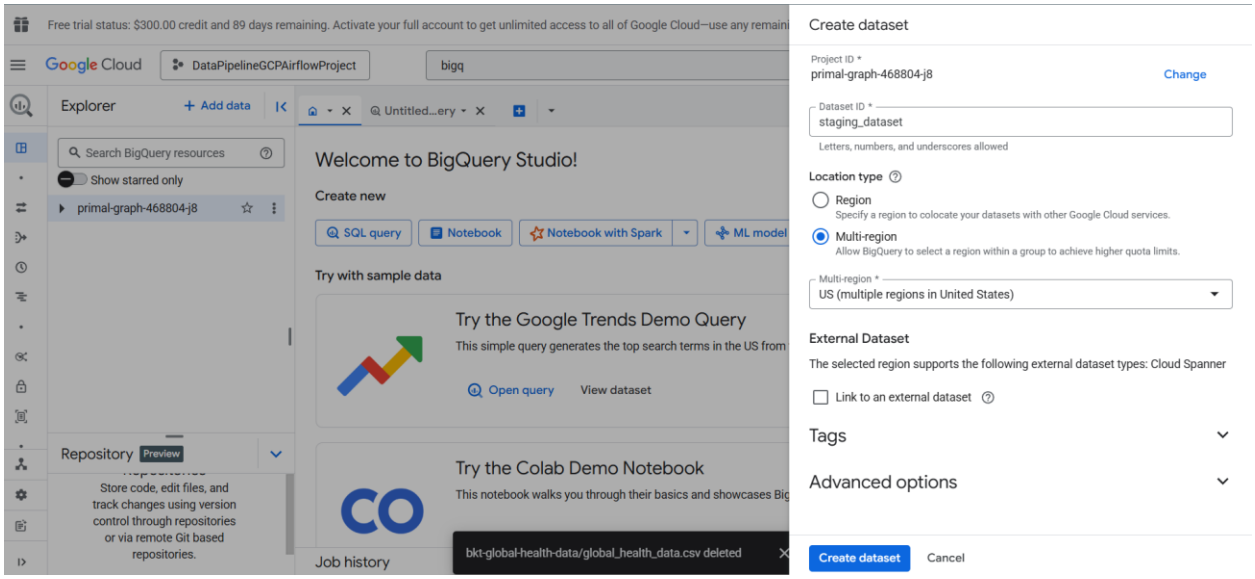
Start Airflow standalone

airflow standalone

Extraction and Loading

9. Creating a STAGING Dataset in bigquery

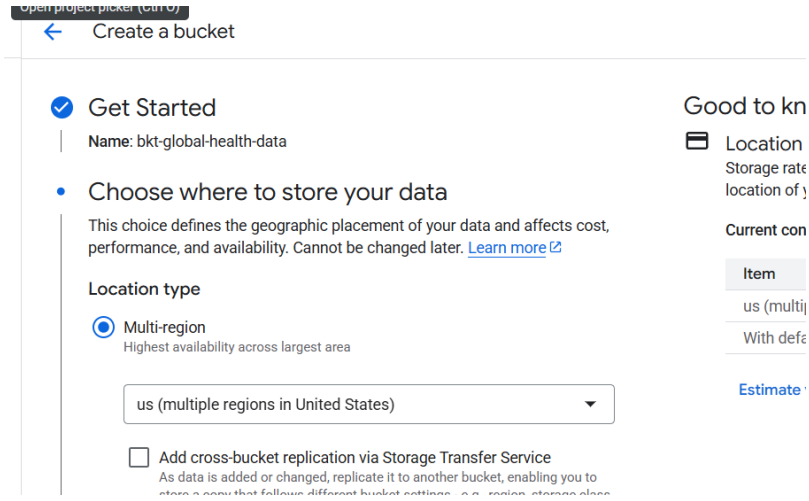
- On the search bar type bigquery



This dataset name will be used as the destination dataset name in the DAG (GCS to BQ)

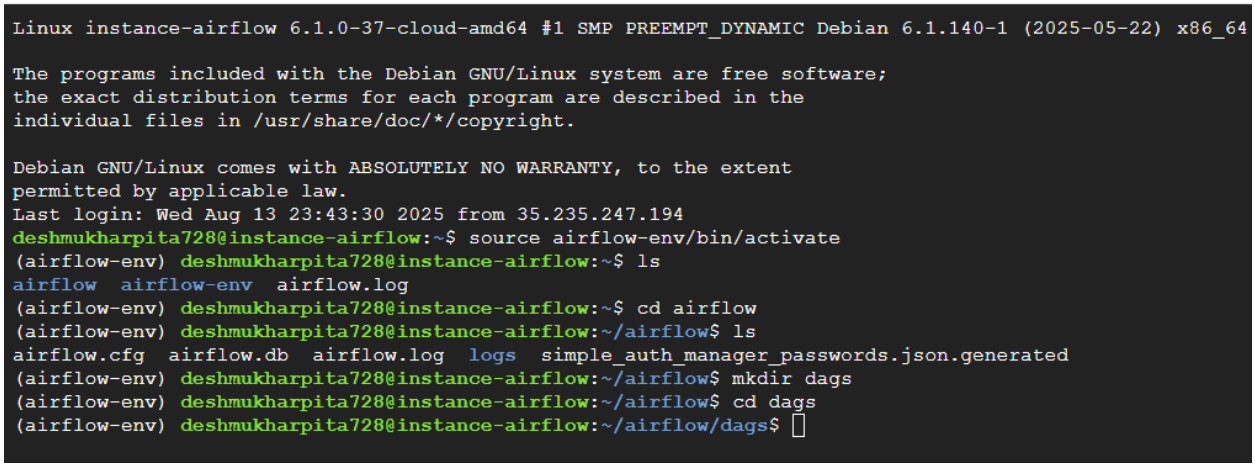
10. Getting the Data into GCP
    a. First create a Cloud Storage Bucket and that will be picked by the airflow DAG

    Name the bucket and hit create > hit confirm



11. Create your FIRST DAG –Directed Acyclic Graph
    This DAG will pick the data from CSV file (in the Google Cloud Storage) and load it to BigQuery (Data Warehouse)

12. Create DAG folder accordingly in the SSH terminal
    dags_folder = /home/deshmukharpita728/airflow/dags



13. Create your first DAG file under the dag folder using the below command, and paste the dag query, hit save and close the file. It should then show up in the Airflow if there are no syntax errors

    nano csv_to_bq.py

```
from airflow.decorators import dag, task from
airflow.providers.google.cloud.transfers.gcs_to_bigquery import GCSToBigQueryOperator
from datetime import datetime

@dag(start_date=datetime(2025, 1, 1), catchup=False, schedule=None,
tags=["csv","bigquery"])

def dag_gcs_to_bquery():

#task to load csv from GCS to Bigquery

load_csv_delimiter = GCSToBigQueryOperator(
task_id="tsk_gcs_to_bigquery",
bucket="bkt-global-health-data",
source_objects=["global_health_data.csv"],
```

```
source_format="CSV",
destination_project_dataset_table='primal-graph-468804-
j8.staging_dataset.global_health_data',
write_disposition="WRITE_TRUNCATE",
skip_leading_rows=1,
field_delimiter=",",
autodetect=True

    )
```

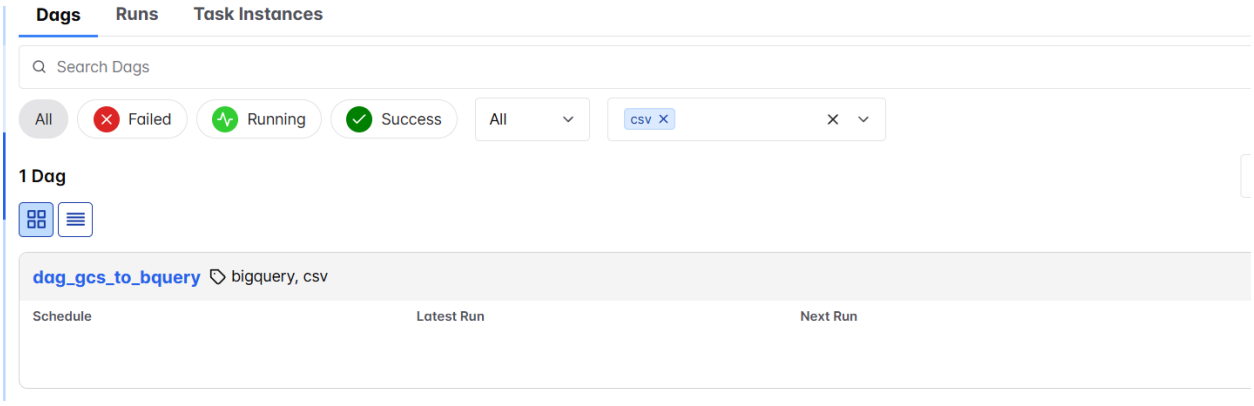
dag_gcs_to_bquery()

```
(airflow-env) deshmukharpita728@instance-airflow:~/airflow$ cd dags
(airflow-env) deshmukharpita728@instance-airflow:~/airflow/dags$ nano csv_to_bq.py
(airflow-env) deshmukharpita728@instance-airflow:~/airflow/dags$ ls
csv_to_bq.py
(airflow-env) deshmukharpita728@instance-airflow:~/airflow/dags$
```

**NOTE: Might need to pip install the google providers

```
(airflow-env) deshmukharpita728@instance-airflow:~/airflow/dags$ pip install apache-airflow-providers-google
Collecting apache-airflow-providers-google
  Downloading apache_airflow_providers_google-17.1.0-py3-none-any.whl (1.0 MB)
```

DAG should show up like this on the web server



**ERROR

The DAG failed with an error conn_id is missing running the below query in the SSH terminal helped to solve the error

1. **Stay in your virtualenv** (looks like you are, `(airflow-env)` is active).
2. Run the command to create `google_cloud_default`:

bash Copy Edit

```
airflow connections add 'google_cloud_default' \
    --conn-type 'google_cloud_platform' \
    --conn-extra '{"use_google_cloud_default": true}'
```

3. Check it was created:

bash Copy Edit

```
airflow connections list
```

You should see `google_cloud_default` in the list.

4. **Restart your Airflow scheduler and webserver** (if already running):

****Remember that we did not upload any CSV file to the Bucket – so the DAG will fail with a file not found error, we don't want the DAG to fail when a file is not found so we will build a sensor

14. Creating the Sensor – the DAG should wait until a file is present in the bucket and then only run

    Updated DAG below

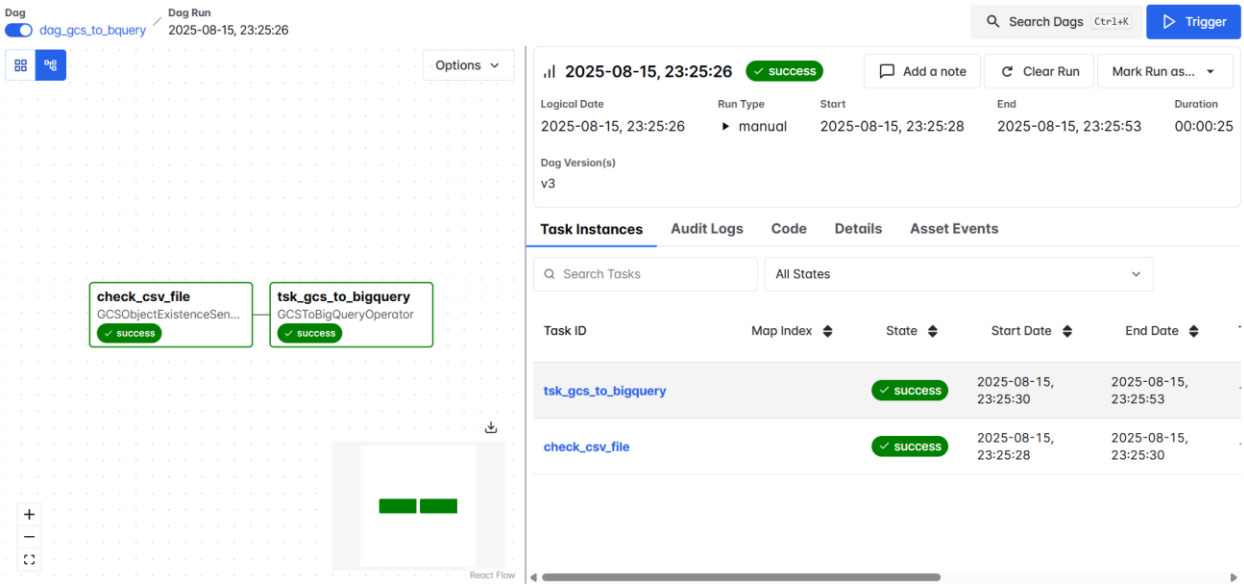
```
from airflow.decorators import dag, task
from airflow.providers.google.cloud.transfers.gcs_to_bigquery import
GCSToBigQueryOperator
from airflow.providers.google.cloud.sensors.gcs import GCSObjectExistenceSensor
from datetime import datetime

@dag(start_date=datetime(2025, 1, 1),
    catchup=False,
    schedule=None,
    tags=["csv","bigquery"])


def dag_gcs_to_bquery():
```

```
    # task to check if csv file is present in GCS

    check_csv_file = GCSObjectExistenceSensor(
    task_id = "check_csv_file",
    bucket = "bkt-global-health-data",
    object="global_health_data.csv",
    poke_interval = 30, #in how many seconds will it look or a file again
    timeout = 300 #after 300s the task will fail
    )



  #task to load csv from GCS to Bigquery

    load_csv_delimiter = GCSToBigQueryOperator(
    task_id="tsk_gcs_to_bigquery",
    bucket="bkt-global-health-data",
    source_objects=["global_health_data.csv"], #this is where file is on GCS
    source_format="CSV",
    destination_project_dataset_table='primal-graph-468804-
j8.staging_dataset.global_health_data',
    write_disposition="WRITE_TRUNCATE",
    skip_leading_rows=1,
    field_delimiter=",",
    autodetect=True

        )

    check_csv_file>>load_csv_delimiter
dag_gcs_to_bquery()
```

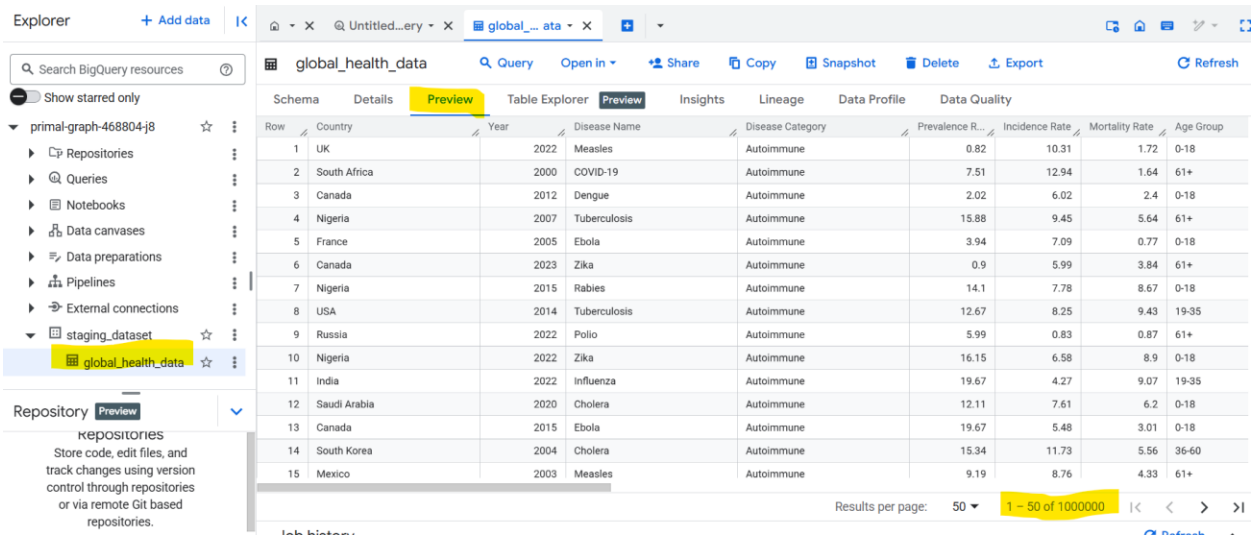
15. This is how the DAG looks like in Airflow graph view

**NOTE: Name of the task shows up on the graph view

16. Successful DAG run – 1M records have been loaded in 24s



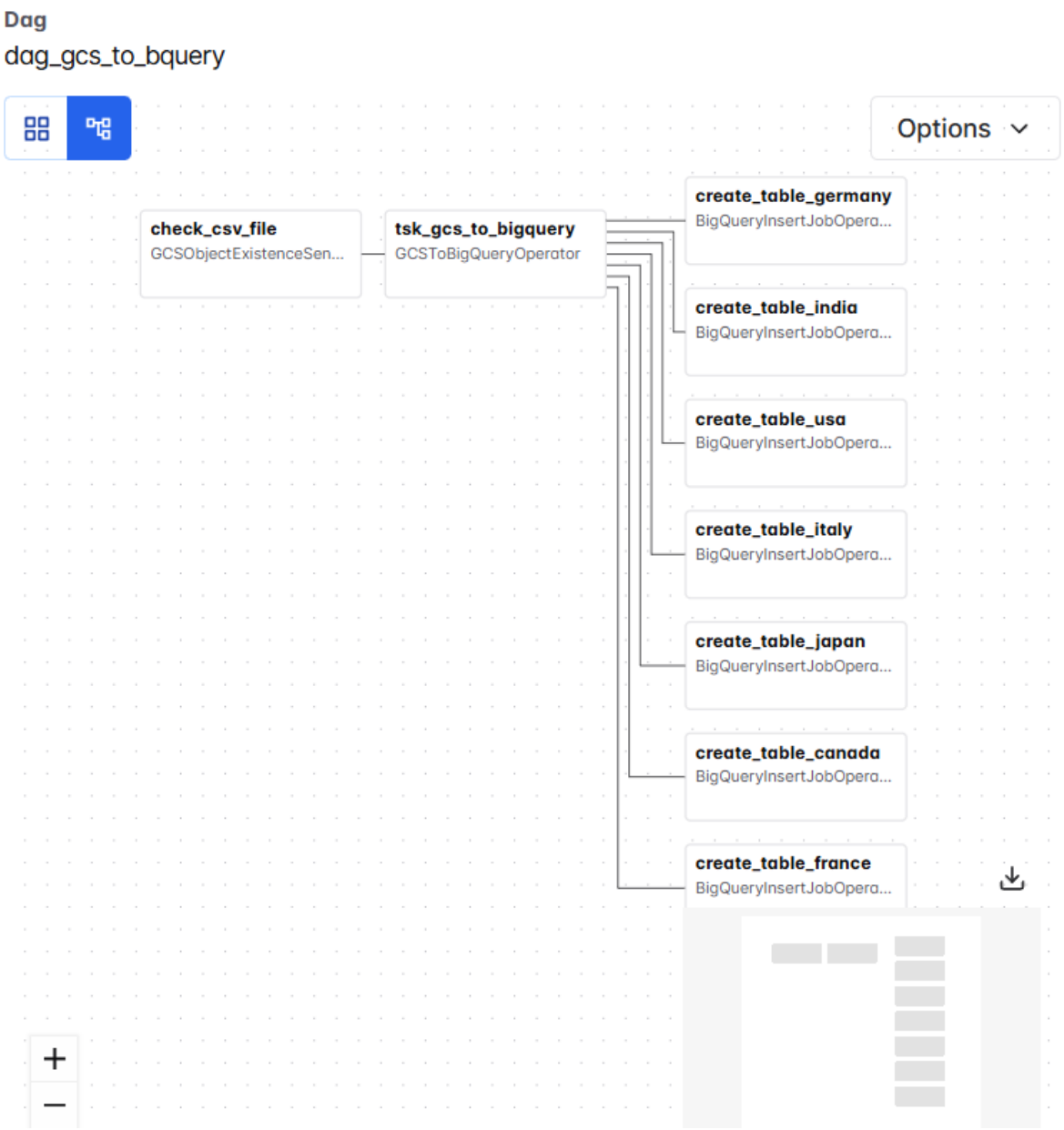
17. Let's validate if 1M rows have been added to BigQuery – search Bigquery on the top search bar



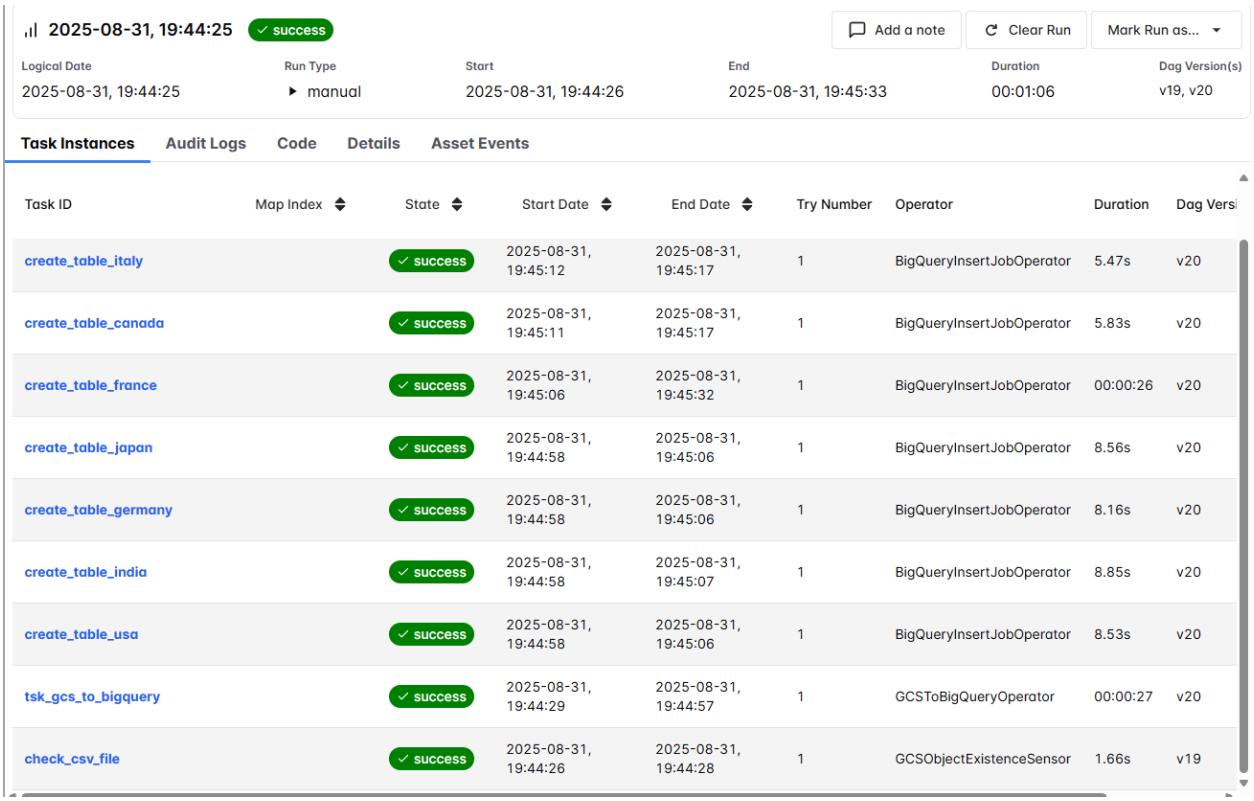
Transformation – In this stage, need to create separate tables for each country

18. Make additions to the existing py file – transformations.py

This is how the updated DAG looks like

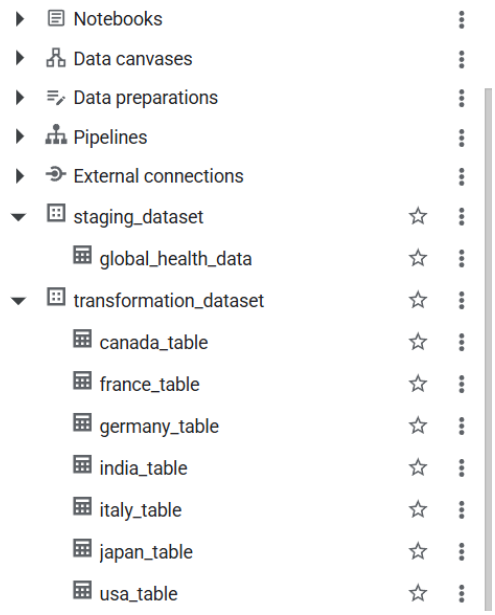


19. The Dag has finished running

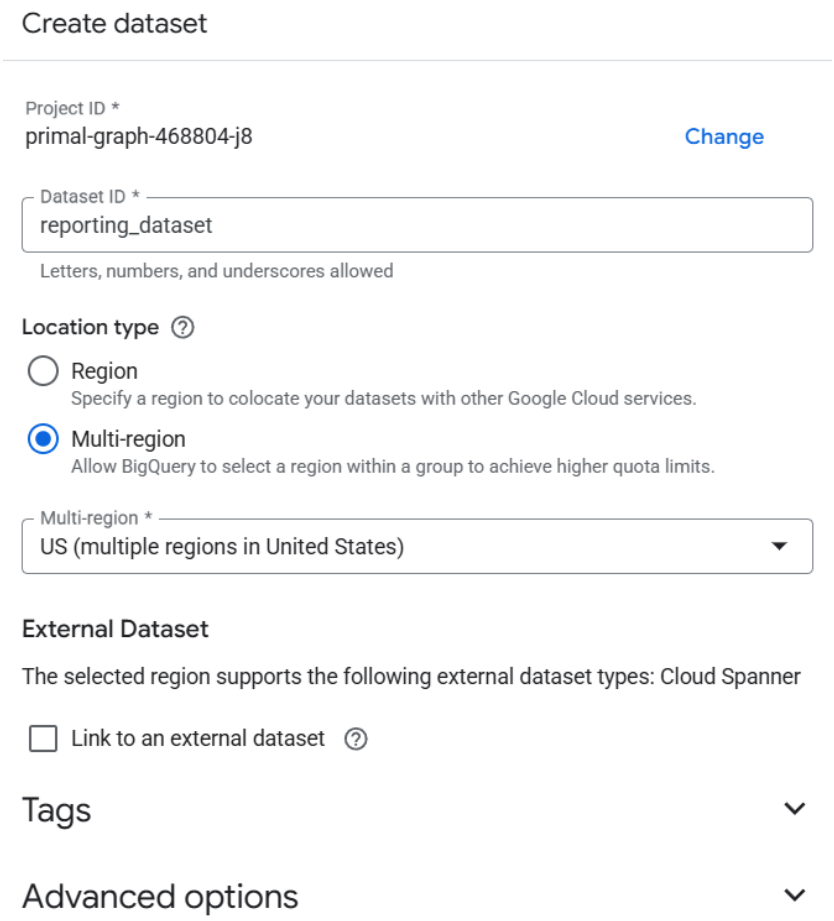


Task ID	Map Index	State	Start Date	End Date	Try Number	Operator	Duration	Dag Versi
create_table_italy		success	2025-08-31, 19:45:12	2025-08-31, 19:45:17	1	BigQueryInsertJobOperator	5.47s	v20
create_table_canada		success	2025-08-31, 19:45:11	2025-08-31, 19:45:17	1	BigQueryInsertJobOperator	5.83s	v20
create_table_france		success	2025-08-31, 19:45:06	2025-08-31, 19:45:32	1	BigQueryInsertJobOperator	00:00:26	v20
create_table_japan		success	2025-08-31, 19:44:58	2025-08-31, 19:45:06	1	BigQueryInsertJobOperator	8.56s	v20
create_table_germany		success	2025-08-31, 19:44:58	2025-08-31, 19:45:06	1	BigQueryInsertJobOperator	8.16s	v20
create_table_india		success	2025-08-31, 19:44:58	2025-08-31, 19:45:07	1	BigQueryInsertJobOperator	8.85s	v20
create_table_usa		success	2025-08-31, 19:44:58	2025-08-31, 19:45:06	1	BigQueryInsertJobOperator	8.53s	v20
tsk_gcs_to_bigquery		success	2025-08-31, 19:44:29	2025-08-31, 19:44:57	1	GCSToBigQueryOperator	00:00:27	v20
check_csv_file		success	2025-08-31, 19:44:26	2025-08-31, 19:44:28	1	GCSObjectExistenceSensor	1.66s	v19

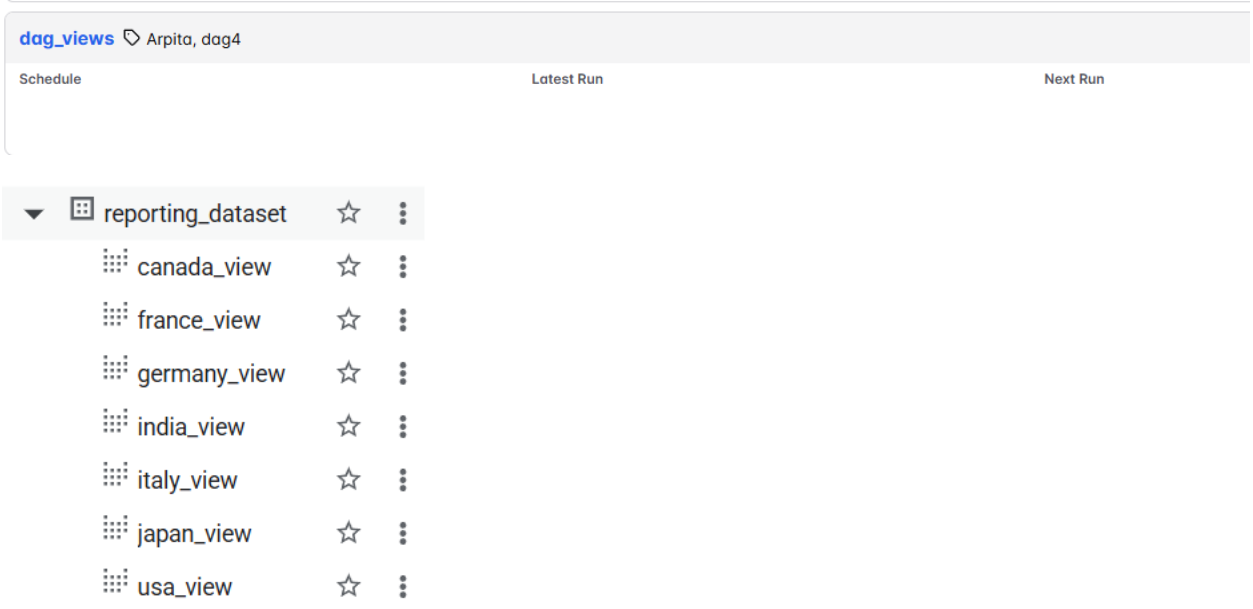
20. Tables get created in the transformation data set



21. Now for reporting purposes, if not all columns from the base tables are need, views can be created to only pull the required columns – Create a Reporting data set



22. Create a DAG to create views –created dag_views.py file



Job Information for running data in CSV vs Table vs View

Select * from global health data where country = ‘India’

CSV

```
1  SELECT * FROM `primal-graph-468804-j8.staging_dataset.global_health_data`
2  where country = 'India'
```

Query completed

Query results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Job ID	primal-graph-468804-j8:US.bquxjob_4b0fd5aa_1990711ce35
User	deshmukharpita728@gmail.com
Location	US
Creation time	Sep 1, 2025, 3:57:11 PM UTC-5
Start time	Sep 1, 2025, 3:57:11 PM UTC-5
End time	Sep 1, 2025, 3:57:12 PM UTC-5
Duration	1 sec
Bytes processed	167.56 MB
Bytes billed	168 MB
Slot milliseconds	1510
Job priority	INTERACTIVE
Use legacy SQL	false
Destination table	Temporary table
Index Usage Mode	UNUSED

Job history

Table

```
1  SELECT * FROM `primal-graph-468804-j8.transformation_dataset.india_table`
```

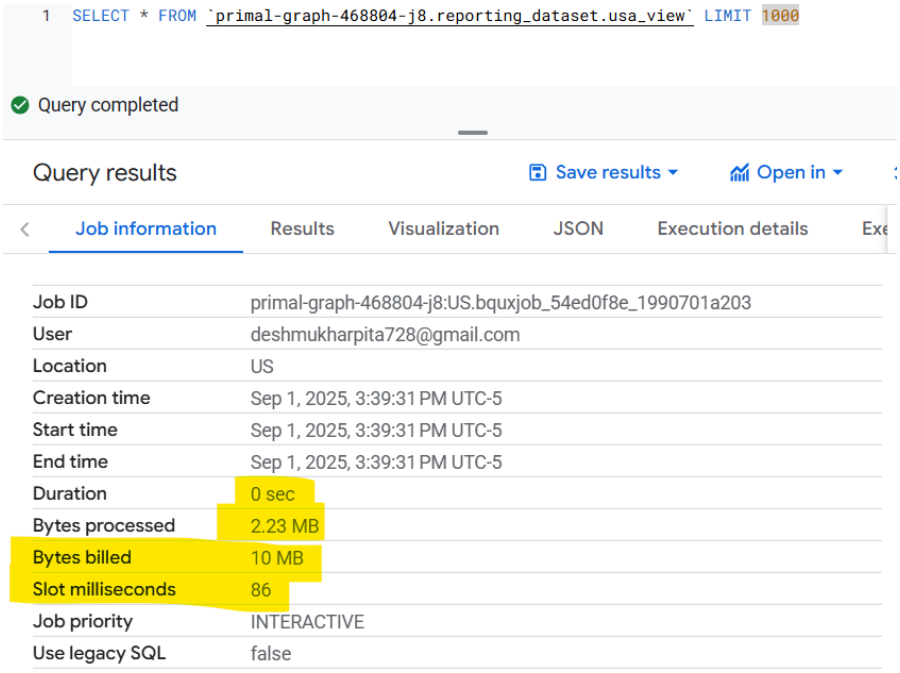
Query completed

Query results

Job information | Results | Visualization | JSON | Execution details | Execut

Job ID	primal-graph-468804-j8:US.bquxjob_3902ccea_19907169fb2
User	deshmukharpita728@gmail.com
Location	US
Creation time	Sep 1, 2025, 4:02:27 PM UTC-5
Start time	Sep 1, 2025, 4:02:27 PM UTC-5
End time	Sep 1, 2025, 4:02:28 PM UTC-5
Duration	1 sec
Bytes processed	8.25 MB
Bytes billed	10 MB
Slot milliseconds	885
Job priority	INTERACTIVE
Use legacy SQL	false
Destination table	Temporary table
Labels	

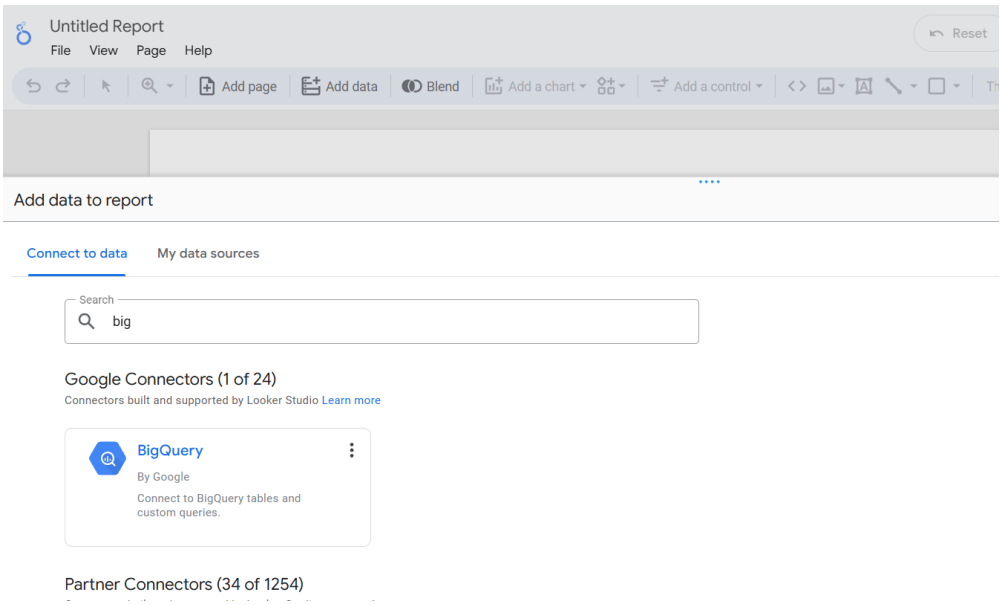
View



Creating a BI report based on the data

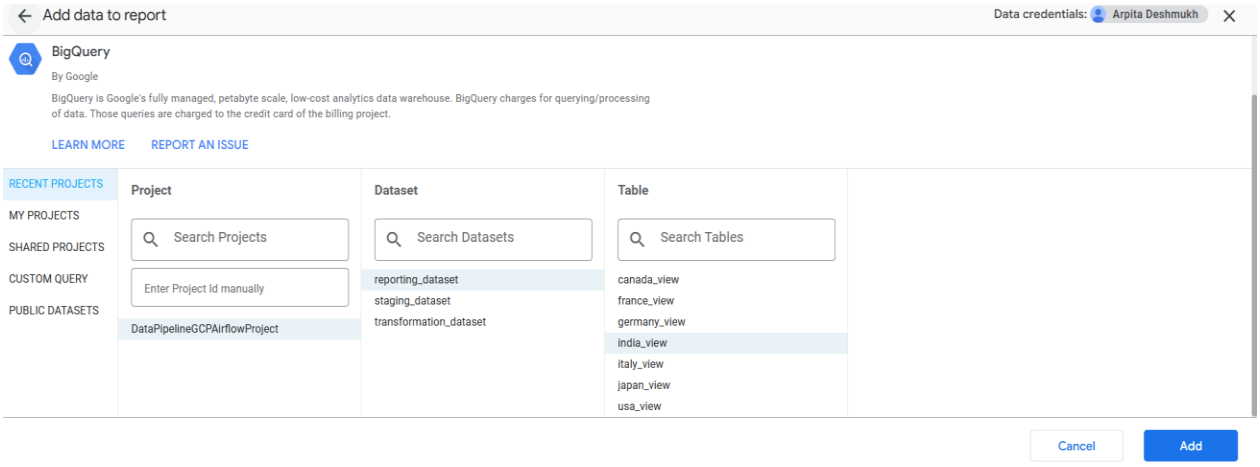
23. Go to Lookerstudio.google.com

24. Create a blank report and connect it to the **Big query** data source



25.

Select the India View



Below is a quick Summary page based on Incidence and Prevalence Rate

disease_category | disease_name | year

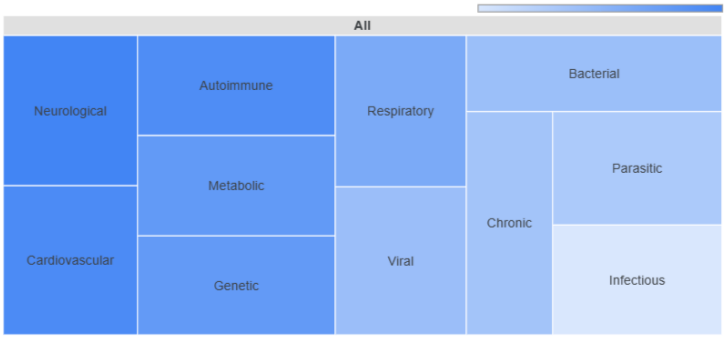
Prevalence - How Common is this disease?
Incidence - How many new people are infected with this disease

Which Diseases Are Spreading at a Faster Rate

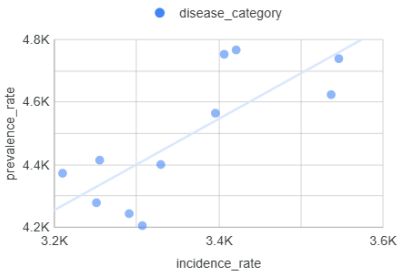
	disease_name	incidence_rate
1.	Parkinson's Disease	9,867.95
2.	Tuberculosis	9,746.05
3.	Cancer	9,732.61
4.	Hepatitis	9,603.21
5.	COVID-19	9,582.77
6.	Influenza	9,551.98
7.	Polio	9,526.84
8.	Zika	9,421.8
9.	Diabetes	9,406.36
10.	Dengue	9,392.19

1 - 10 / 20

Most Prevalent Diseases in India



Do Diseases with Higher Prevalence Also Have Higher Incidence?



***A higher incidence tends to increase prevalence if recovery or death is slow.

Top 3 Diseases that are on the Rise (Incidence Rate)