Apache Airflow assignment

>Basics of Apache airflow -

https://www.youtube.com/watch?v=AHMm1wfGuHE&list=PLYizQ5FvN6pvIOcOd6dFZu3lQqc6zBGp2

>Apache airflow setup:

Link - https://airflow.apache.org/docs/stable/start.html

Steps to be followed:

1. export AIRFLOW_HOME=~/airflow
2. **Installing airflow via pip** - pip install apache-airflow
3. **Initialize the database** - airflow initdb
4. **Start the webserver** - airflow webserver -p 8080
5. **Start the scheduler** - airflow scheduler

>BigQuery Setup

Link - https://cloud.google.com/bigquery/docs/quickstarts/quickstart-client-libraries

Steps to be followed:

1. **In the Cloud Console, on the project selector page, select or create a Cloud project.**
   Note: If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.
   Go to the project selector page
2. **Enable the BigQuery API.**
   Enable the API
3. **Set up authentication:**
   1. In the Cloud Console, go to the Create service account key page.

      Go to the Create Service Account Key page

   2. From the Service account list, select New service account.

   3. In the Service account name field, enter a name.

4. From the Role list, select Project > Owner.

4. Set the environment variable GOOGLE_APPLICATION_CREDENTIALS to the path of the JSON file that contains your service account key. This variable only applies to your current shell session, so if you open a new session, set the variable again.

5. Create database id and table id with proper schema.

6. Setup complete.

CODEBASE-

>Steps to make a dag:

1. **Import the modules.**
2. **List out the default arguments.**
3. **Initialize the dag.**
4. **Define the tasks.**
5. **Prioritize the tasks.**

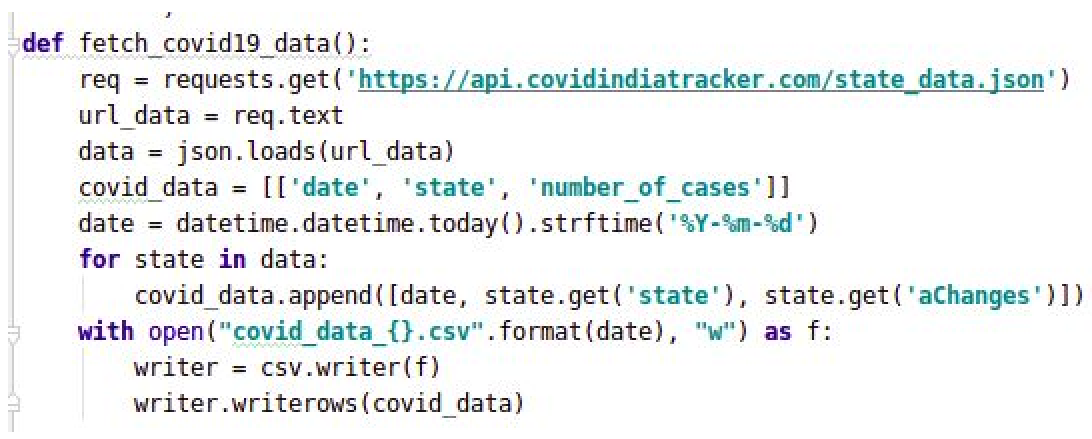
Tasks required for the DAG:

Task 1:

1. Fetch the covid19 data of the states using the following api-

   https://api.covidindiatracker.com/state_data.json

2. Write the following data to a csv.



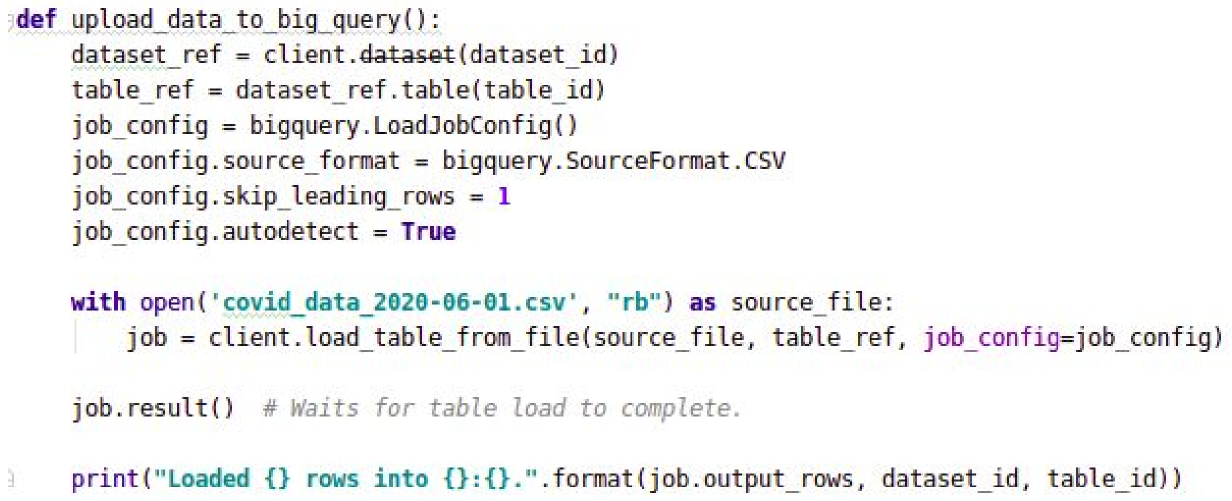
```python
def fetch_covid19_data():
    req = requests.get('https://api.covidindiatracker.com/state_data.json')
    url_data = req.text
    data = json.loads(url_data)
    covid_data = [['date', 'state', 'number_of_cases']]
    date = datetime.datetime.today().strftime('%Y-%m-%d')
    for state in data:
        covid_data.append([date, state.get('state'), state.get('aChanges')])
    with open("covid_data_{}.csv".format(date), "w") as f:
        writer = csv.writer(f)
        writer.writerows(covid_data)
```

t1 = PythonOperator(task_id='fetch_data', python_callable=fetch_covid19_data, dag=dag)

Task 2:

This task assists in uploading the data from the local data source i.e. CSV to a BigQuery table.

In order to load the data to bigquery, we need to setup the project in the google console and download the json file which contains the credentials.

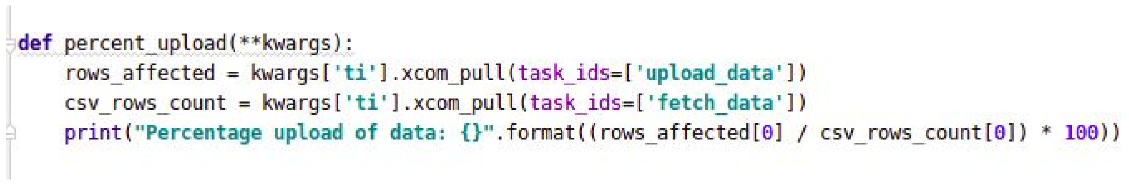


```python
def upload_data_to_big_query():
    dataset_ref = client.dataset(dataset_id)
    table_ref = dataset_ref.table(table_id)
    job_config = bigquery.LoadJobConfig()
    job_config.source_format = bigquery.SourceFormat.CSV
    job_config.skip_leading_rows = 1
    job_config.autodetect = True

    with open('covid_data_2020-06-01.csv', "rb") as source_file:
        job = client.load_table_from_file(source_file, table_ref, job_config=job_config)

    job.result()  # Waits for table load to complete.

    print("Loaded {} rows into {}:{}.".format(job.output_rows, dataset_id, table_id))
```

t2 = PythonOperator(task_id='upload_data', python_callable=upload_data_to_big_query, dag=dag)

Task 3:

This task is used to read the data (no. of rows) from the BigQuery table and flag a status saying percentage of upload. (total rows in BQ table for today * 100 / total rows in today’s CSV)

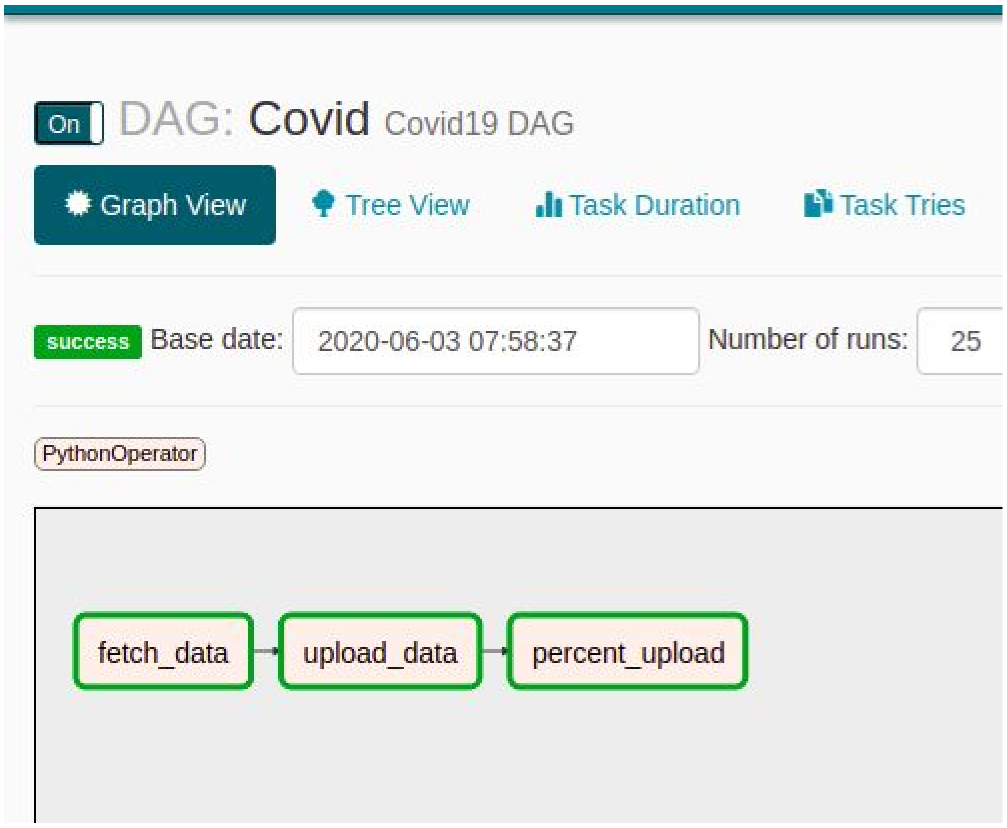


```
def percent_upload(**kwargs):
    rows_affected = kwargs['ti'].xcom_pull(task_ids=['upload_data'])
    csv_rows_count = kwargs['ti'].xcom_pull(task_ids=['fetch_data'])
    print("Percentage upload of data: {}".format((rows_affected[0] / csv_rows_count[0]) * 100))
```

t3 = PythonOperator(task_id='percent_upload', python_callable=percent_upload, provide_context=True, dag=dag)

>Screenshots of the running pipeline -

Fig 1: Graph View:



DAG: Covid Covid19 DAG

success Base date: 2020-06-03 07:58:37 Number of runs: 25

Fig 2: Tree View:

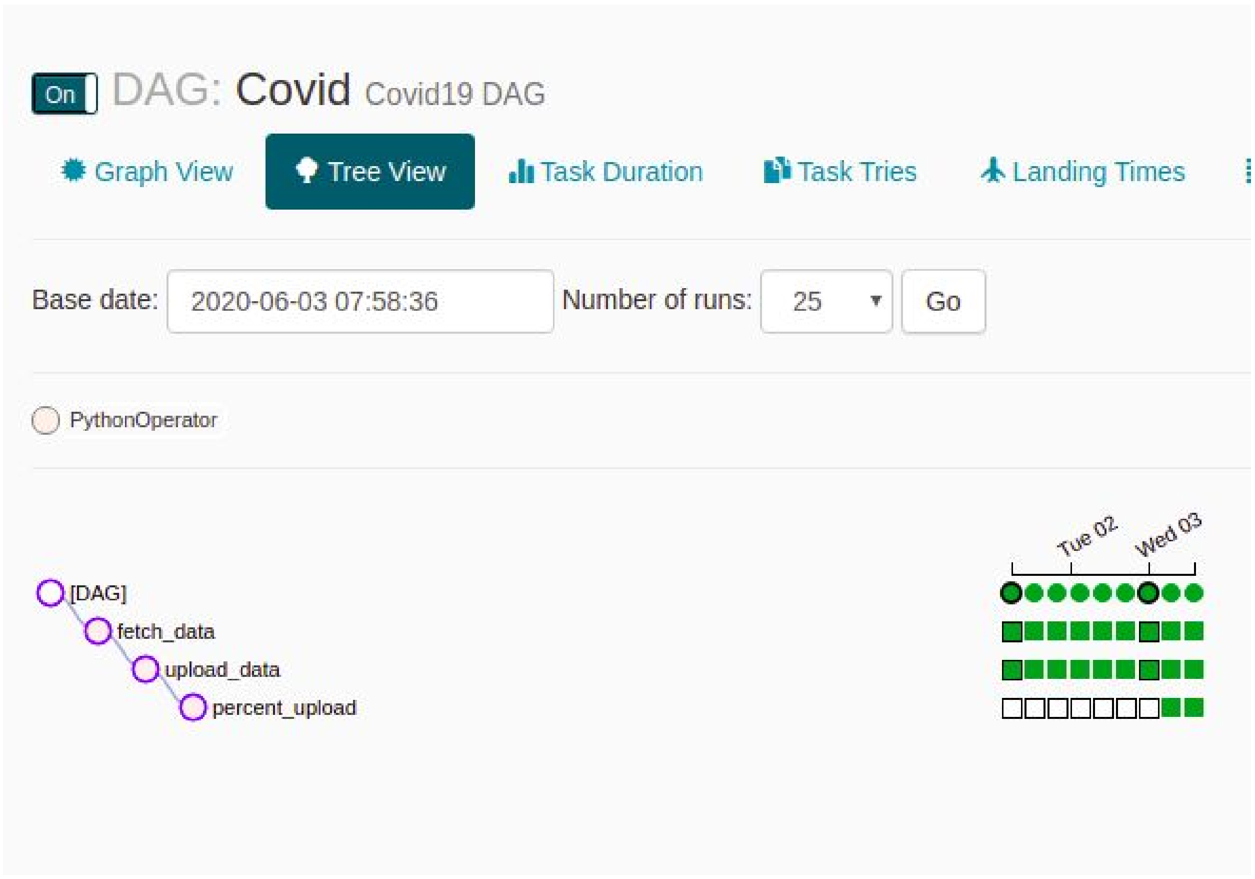
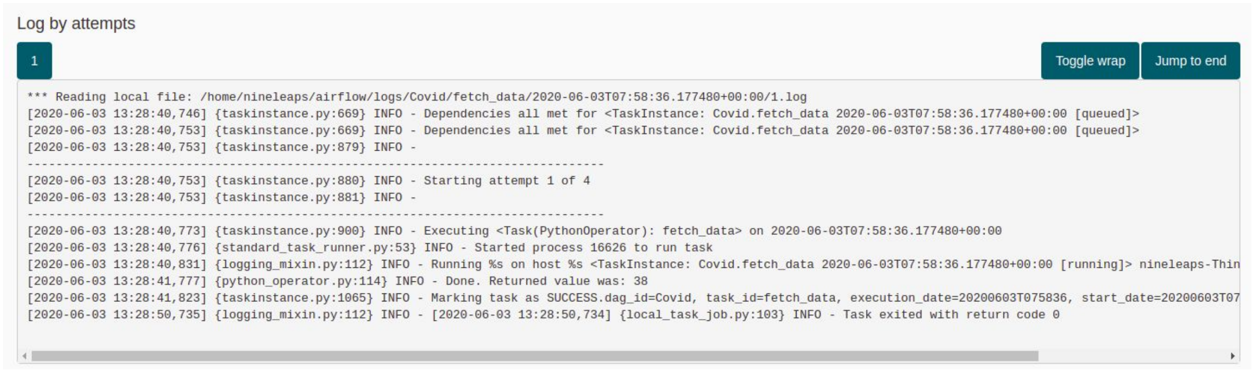


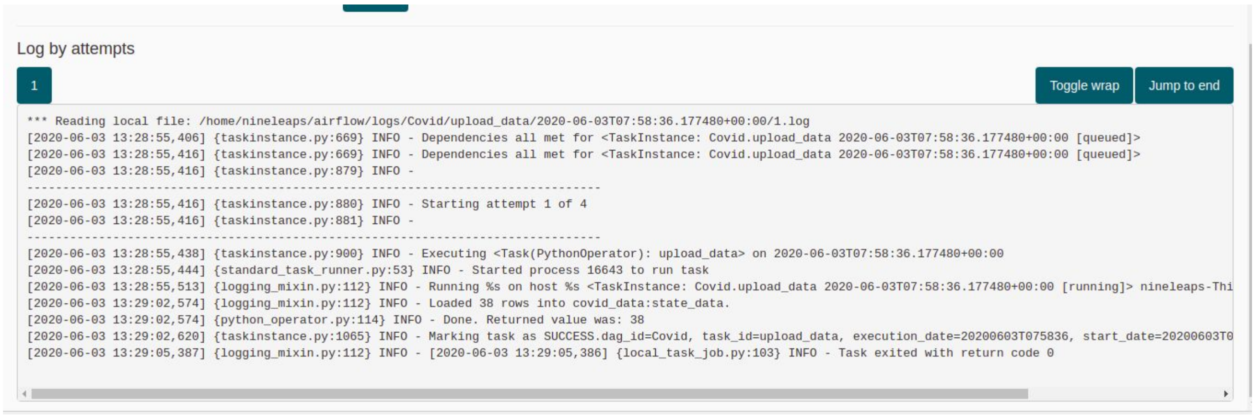
Fig 3: Logs:

Task1:



```
*** Reading local file: /home/nineleaps/airflow/logs/Covid/fetch_data/2020-06-03T07:58:36.177480+00:00/1.log
[2020-06-03 13:28:40,746] {taskinstance.py:669} INFO - Dependencies all met for <TaskInstance: Covid.fetch_data 2020-06-03T07:58:36.177480+00:00 [queued]>
[2020-06-03 13:28:40,753] {taskinstance.py:669} INFO - Dependencies all met for <TaskInstance: Covid.fetch_data 2020-06-03T07:58:36.177480+00:00 [queued]>
[2020-06-03 13:28:40,753] {taskinstance.py:879} INFO -
--------------------------------------------------------------------------------
[2020-06-03 13:28:40,753] {taskinstance.py:880} INFO - Starting attempt 1 of 4
[2020-06-03 13:28:40,753] {taskinstance.py:881} INFO -
--------------------------------------------------------------------------------
[2020-06-03 13:28:40,773] {taskinstance.py:900} INFO - Executing <Task(PythonOperator): fetch_data> on 2020-06-03T07:58:36.177480+00:00
[2020-06-03 13:28:40,776] {standard_task_runner.py:53} INFO - Started process 16626 to run task
[2020-06-03 13:28:40,831] {logging_mixin.py:112} INFO - Running %s on host %s <TaskInstance: Covid.fetch_data 2020-06-03T07:58:36.177480+00:00 [running]> nineleaps-Thin
[2020-06-03 13:28:41,777] {python_operator.py:114} INFO - Done. Returned value was: 38
[2020-06-03 13:28:41,823] {taskinstance.py:1065} INFO - Marking task as SUCCESS.dag_id=Covid, task_id=fetch_data, execution_date=20200603T075836, start_date=20200603T07
[2020-06-03 13:28:50,735] {logging_mixin.py:112} INFO - [2020-06-03 13:28:50,734] {local_task_job.py:103} INFO - Task exited with return code 0
```

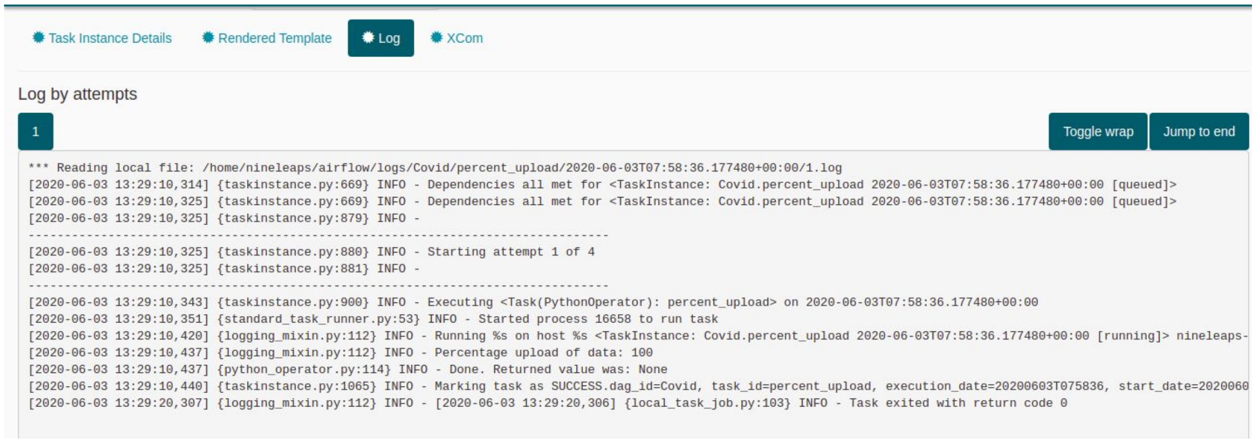
Task 2:


Log by attempts

1 Toggle wrap Jump to end

```
*** Reading local file: /home/nineleaps/airflow/logs/Covid/upload_data/2020-06-03T07:58:36.177480+00:00/1.log
[2020-06-03 13:28:55,406] {taskinstance.py:669} INFO - Dependencies all met for <TaskInstance: Covid.upload_data 2020-06-03T07:58:36.177480+00:00 [queued]>
[2020-06-03 13:28:55,416] {taskinstance.py:669} INFO - Dependencies all met for <TaskInstance: Covid.upload_data 2020-06-03T07:58:36.177480+00:00 [queued]>
[2020-06-03 13:28:55,416] {taskinstance.py:879} INFO -
--------------------------------------------------------------------------------
[2020-06-03 13:28:55,416] {taskinstance.py:880} INFO - Starting attempt 1 of 4
[2020-06-03 13:28:55,416] {taskinstance.py:881} INFO -
--------------------------------------------------------------------------------
[2020-06-03 13:28:55,438] {taskinstance.py:900} INFO - Executing <Task(PythonOperator): upload_data> on 2020-06-03T07:58:36.177480+00:00
[2020-06-03 13:28:55,444] {standard_task_runner.py:53} INFO - Started process 16643 to run task
[2020-06-03 13:28:55,513] {logging_mixin.py:112} INFO - Running %s on host %s <TaskInstance: Covid.upload_data 2020-06-03T07:58:36.177480+00:00 [running]> nineleaps-Thi
[2020-06-03 13:29:02,574] {logging_mixin.py:112} INFO - Loaded 38 rows into covid_data:state_data.
[2020-06-03 13:29:02,574] {python_operator.py:114} INFO - Done. Returned value was: 38
[2020-06-03 13:29:02,620] {taskinstance.py:1065} INFO - Marking task as SUCCESS.dag_id=Covid, task_id=upload_data, execution_date=20200603T075836, start_date=20200603T0
[2020-06-03 13:29:05,387] {logging_mixin.py:112} INFO - [2020-06-03 13:29:05,386] {local_task_job.py:103} INFO - Task exited with return code 0
```

Task 3:

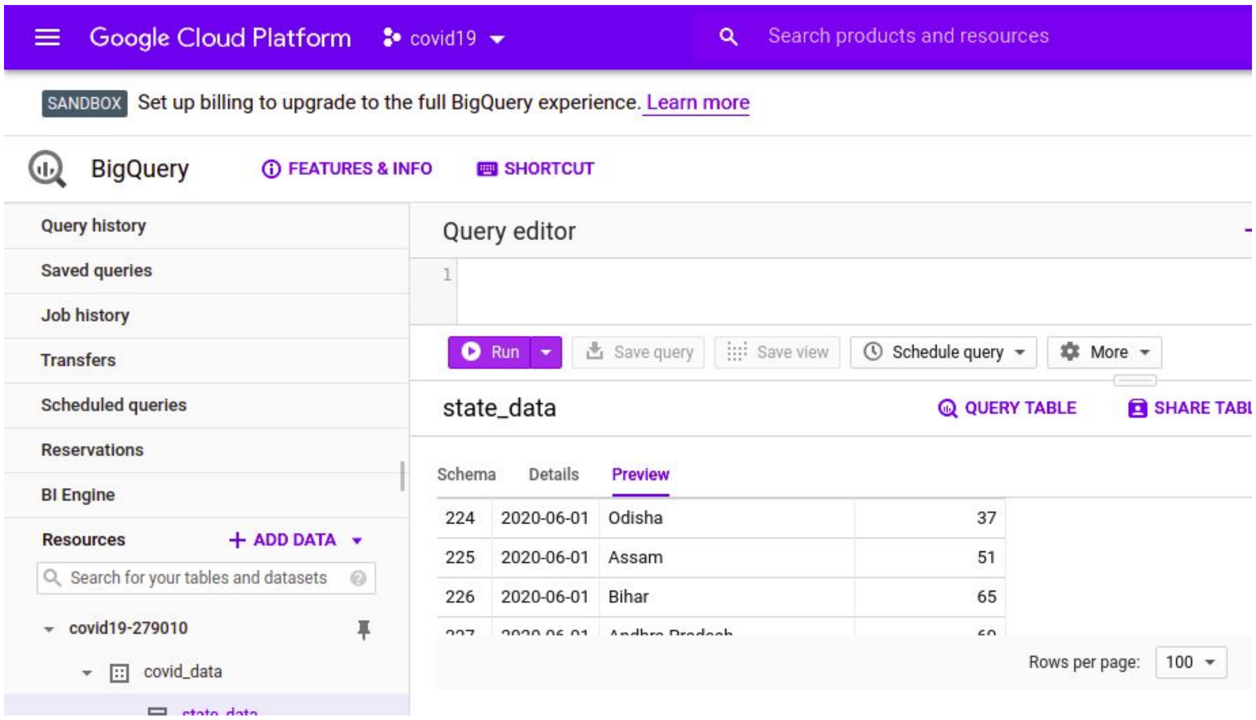


Log by attempts

1 Toggle wrap Jump to end

```
*** Reading local file: /home/nineleaps/airflow/logs/Covid/percent_upload/2020-06-03T07:58:36.177480+00:00/1.log
[2020-06-03 13:29:10,314] {taskinstance.py:669} INFO - Dependencies all met for <TaskInstance: Covid.percent_upload 2020-06-03T07:58:36.177480+00:00 [queued]>
[2020-06-03 13:29:10,325] {taskinstance.py:669} INFO - Dependencies all met for <TaskInstance: Covid.percent_upload 2020-06-03T07:58:36.177480+00:00 [queued]>
[2020-06-03 13:29:10,325] {taskinstance.py:879} INFO -
--------------------------------------------------------------------------------
[2020-06-03 13:29:10,325] {taskinstance.py:880} INFO - Starting attempt 1 of 4
[2020-06-03 13:29:10,325] {taskinstance.py:881} INFO -
--------------------------------------------------------------------------------
[2020-06-03 13:29:10,343] {taskinstance.py:900} INFO - Executing <Task(PythonOperator): percent_upload> on 2020-06-03T07:58:36.177480+00:00
[2020-06-03 13:29:10,351] {standard_task_runner.py:53} INFO - Started process 16658 to run task
[2020-06-03 13:29:10,420] {logging_mixin.py:112} INFO - Running %s on host %s <TaskInstance: Covid.percent_upload 2020-06-03T07:58:36.177480+00:00 [running]> nineleaps-
[2020-06-03 13:29:10,437] {logging_mixin.py:112} INFO - Percentage upload of data: 100
[2020-06-03 13:29:10,437] {python_operator.py:114} INFO - Done. Returned value was: None
[2020-06-03 13:29:10,440] {taskinstance.py:1065} INFO - Marking task as SUCCESS.dag_id=Covid, task_id=percent_upload, execution_date=20200603T075836, start_date=2020060
[2020-06-03 13:29:20,307] {logging_mixin.py:112} INFO - [2020-06-03 13:29:20,306] {local_task_job.py:103} INFO - Task exited with return code 0
```

Fig 4: BigQuery table



224	2020-06-01	Odisha	37
225	2020-06-01	Assam	51
226	2020-06-01	Bihar	65