**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**](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**](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**](https://console.cloud.google.com/projectselector2/home/dashboard)
2. **Enable the BigQuery API.**[**Enable the API**](https://console.cloud.google.com/flows/enableapi?apiid=bigquery)
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](https://console.cloud.google.com/apis/credentials/serviceaccountkey)
   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.**

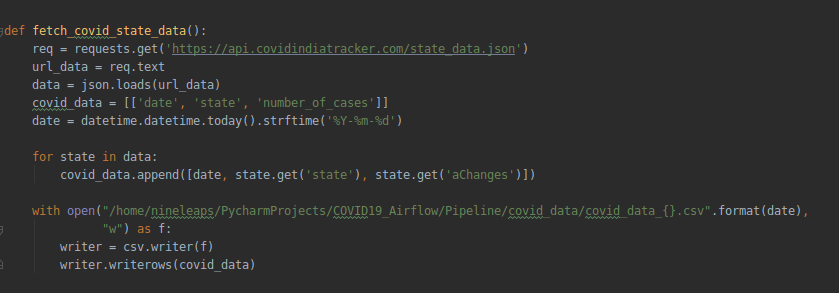
**Tasks required for the DAG:**

**Task 1:**

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

**https://api.covidindiatracker.com/state\_data.json**

1. Write the following data to a csv.

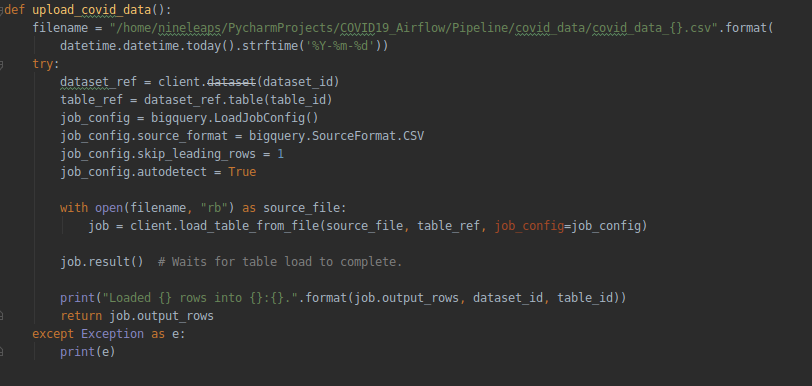
****

**t1 = PythonOperator(task\_id='fetch\_data', python\_callable=fetch\_covid\_state\_data, dag=dag)**

**Task 2:**

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

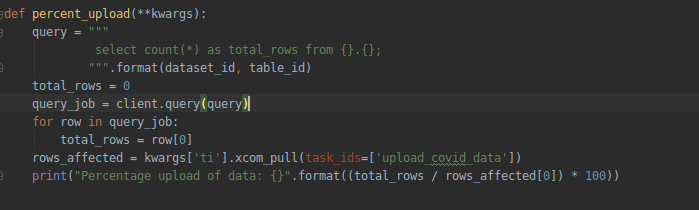
In order to load the data to bigquery, we need to setup the project in the google console and download the json file which contains the credentials.



**t2 = PythonOperator(task\_id='upload\_covid\_data', python\_callable=upload\_covid\_data, dag=dag)**

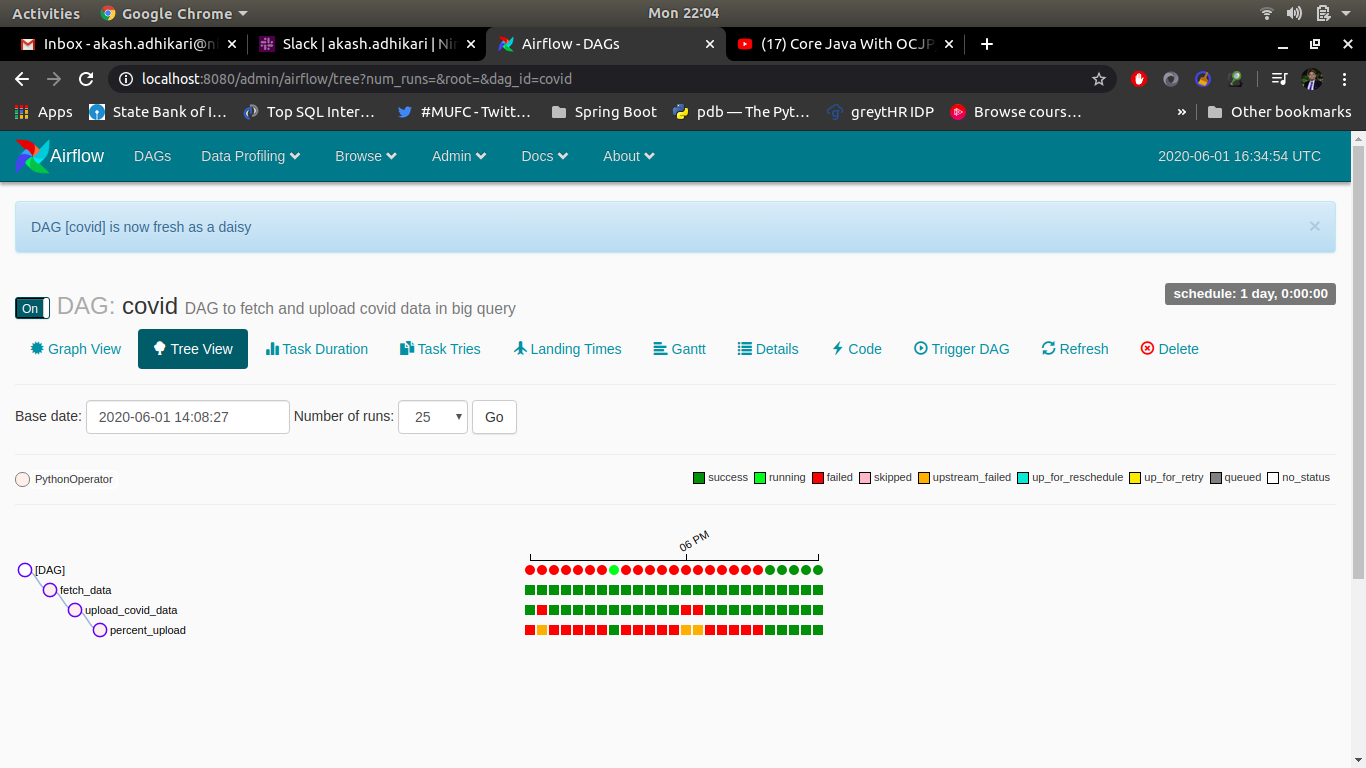
**Task 3:**

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)

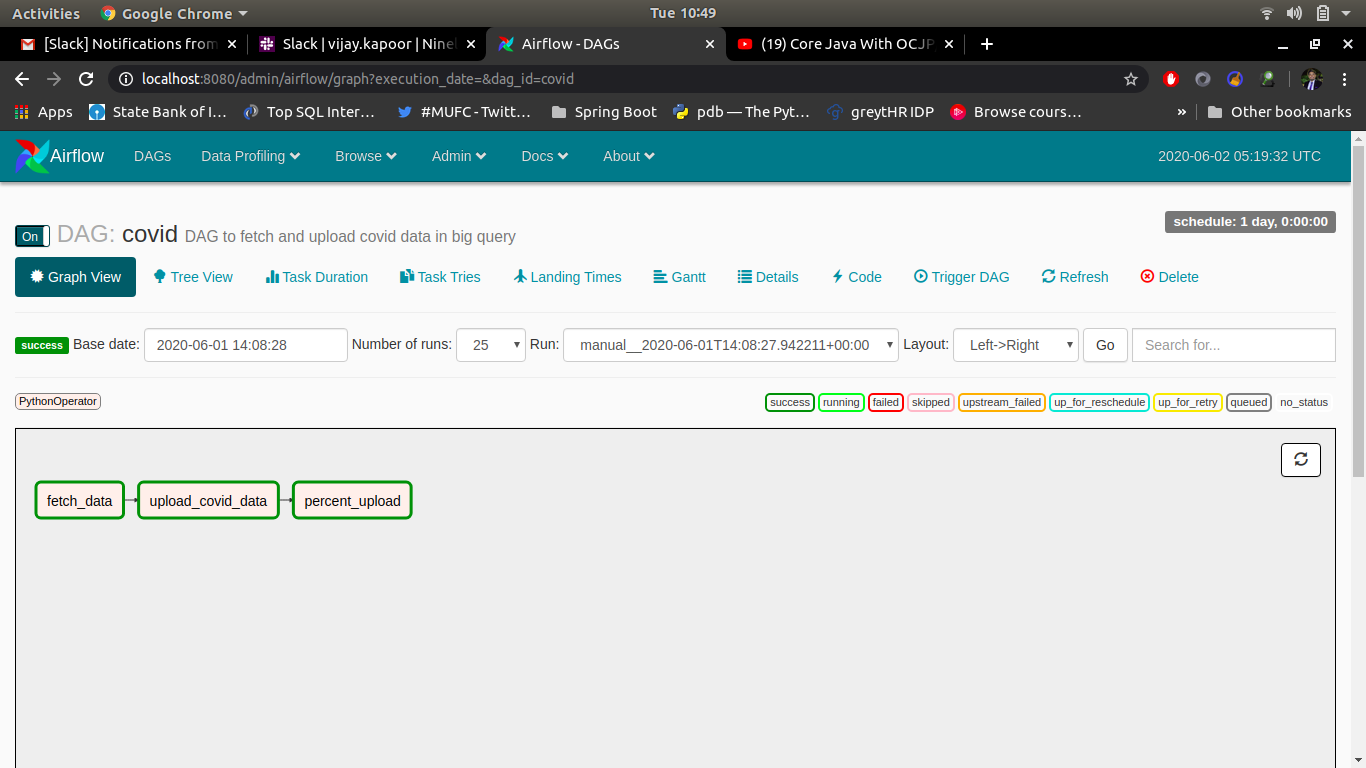
  
  
**t3 = PythonOperator(task\_id='percent\_upload', python\_callable=percent\_upload, provide\_context=True, dag=dag)**

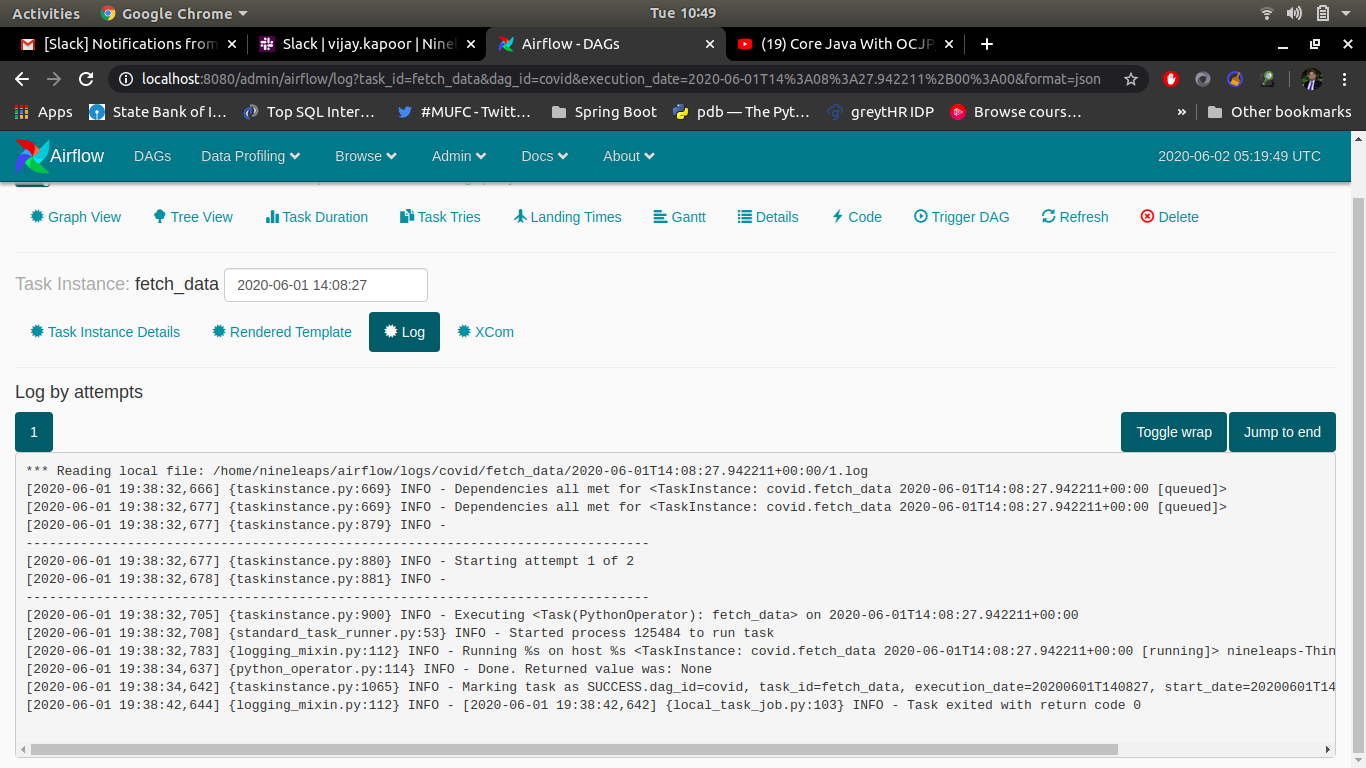
**>Screenshots of the running pipeline -**

**Fig 1: Tree View:**

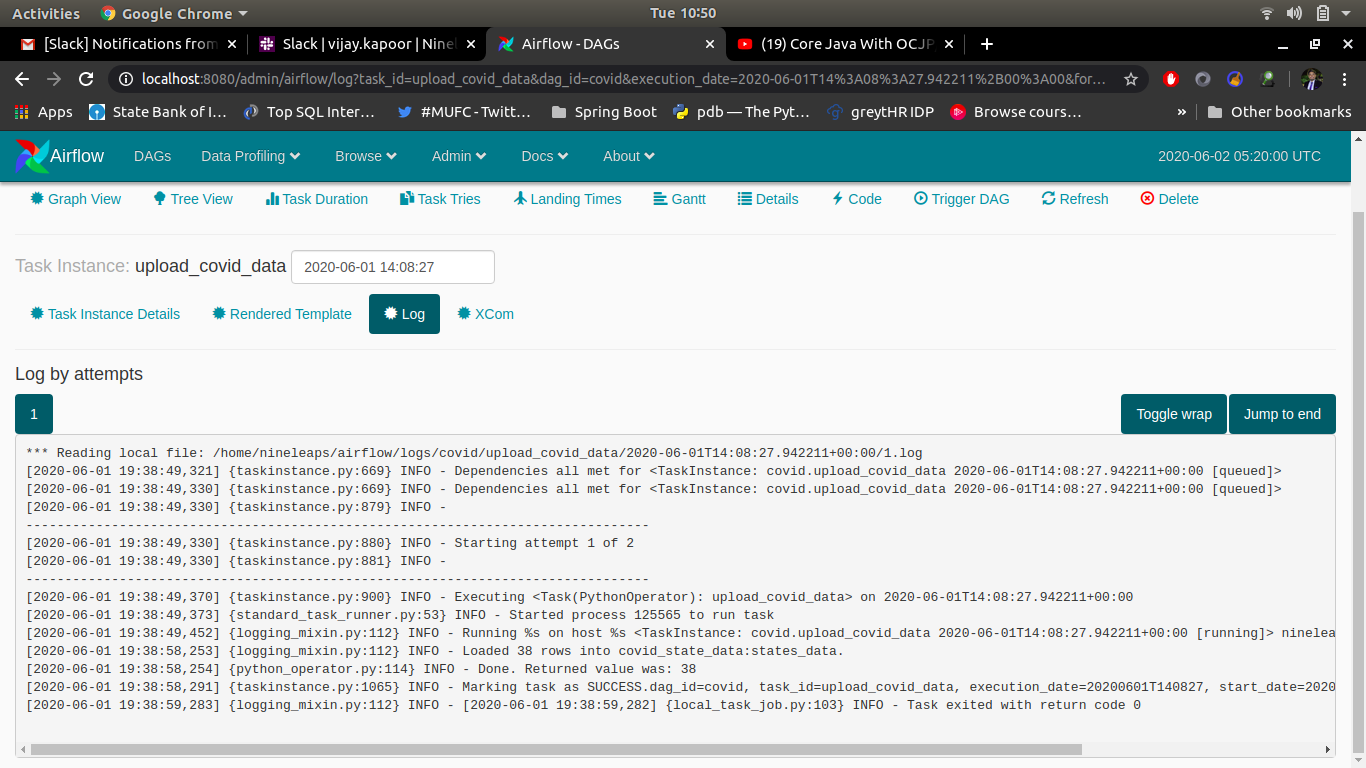
****

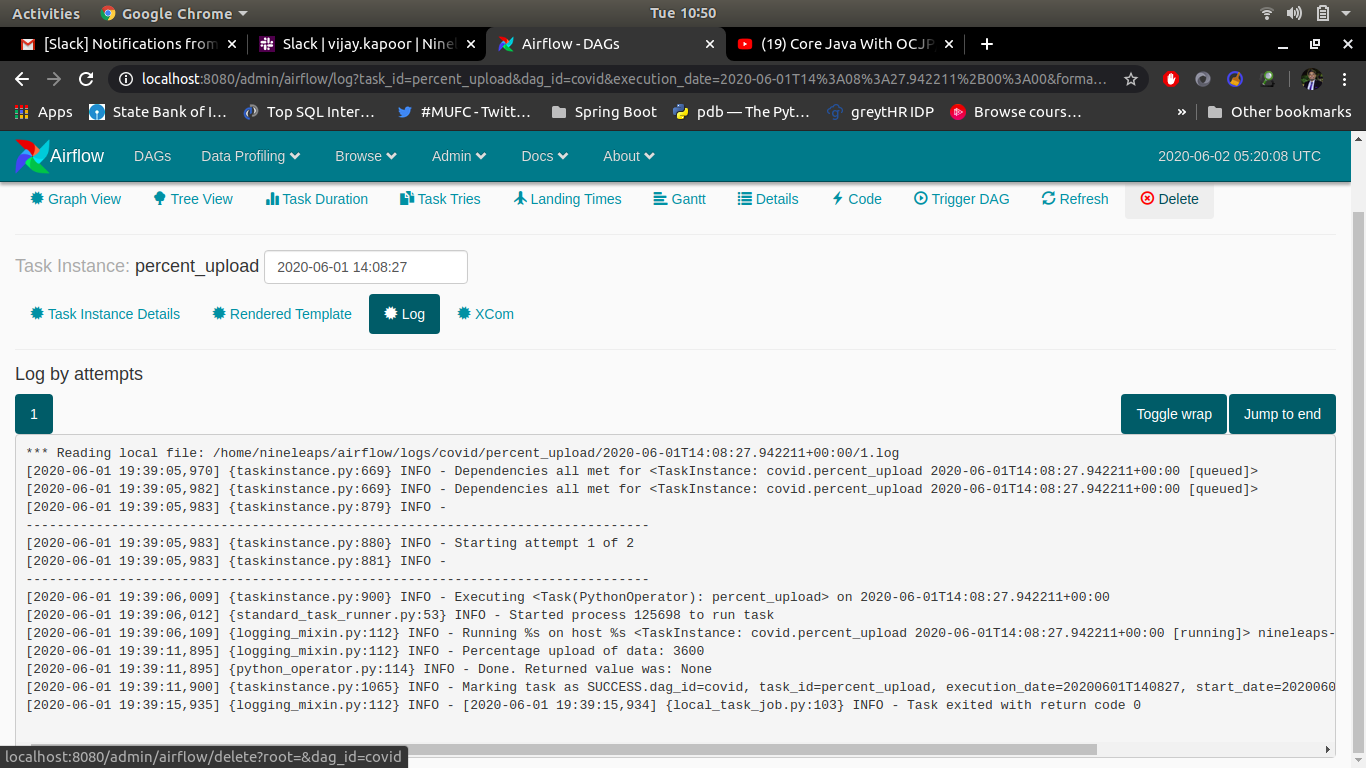
**Fig 2: Graph View:**

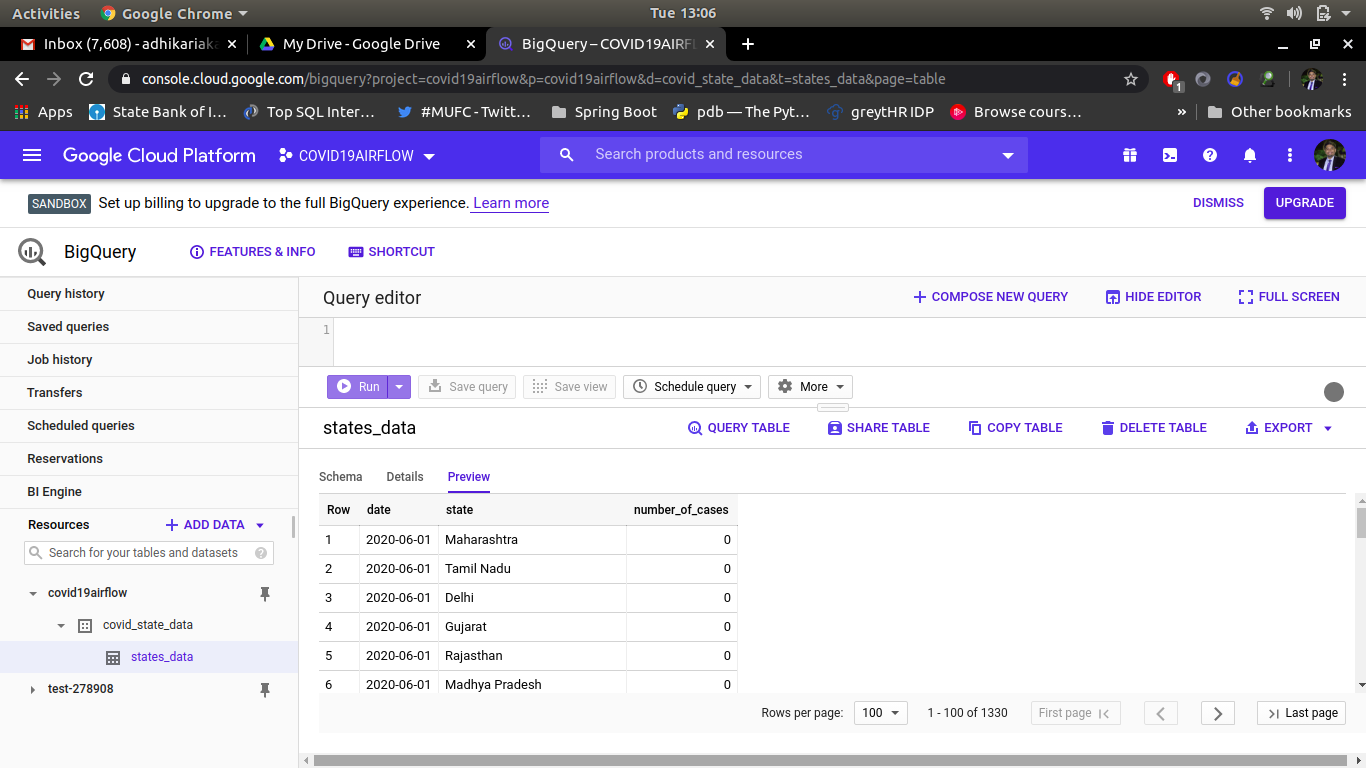
**Fig 3: Logs:  
  
Task1:**

****

**Task 2:**

 **Task 3:**

  
**Big Query Data Upload:**

  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
**Updated code:**

Made the code more dynamic, creation of table can be done by code only, no need to create table from BigQuery UI.

