Name – Puspak Chakraborty

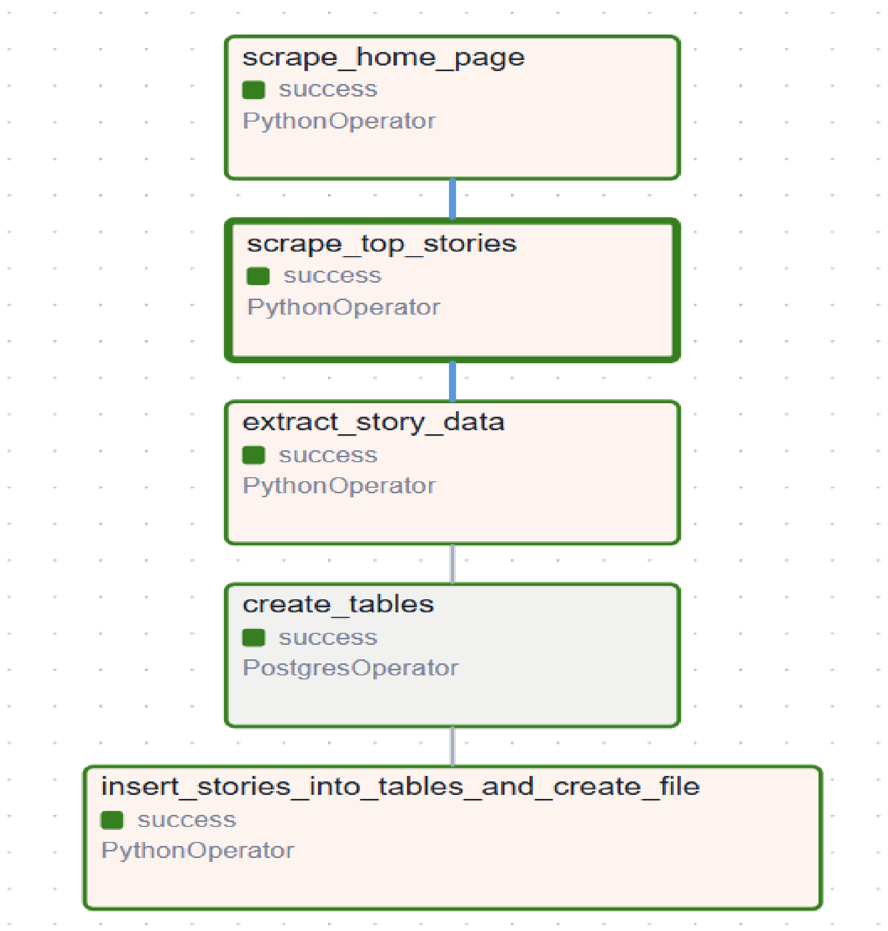
Roll No – DA24S002

Task: Develop an automated system to scrape Google News top stories hourly, store new articles in a database, and send email notifications for new insertions.
System Architecture
The project is divided into two main DAGs (Directed Acyclic Graphs):

1. image_dataset_and_file_creator
2. email_sender_and_file_deleter

DAG 1: image_dataset_and_file_creator
Workflow:

1. Scrape Google News homepage
2. Navigate to and scrape Top Stories page, handling lazy loading
3. Extract headlines and images from top stories
4. Create database tables if not existing
5. Insert new stories into the database
6. Create a status file with the count of new insertions



For the task a, b, c, we are using the functionality from the python code written in the previous assignment. We are using PythonOperator to call those python files.

(The python files are present in the following locations:

/opt/airflow/include/scraper.py,

/opt/airflow/include/extractor.py,

/opt/airflow/include/config.py,

)

For passing the scraped html file from first state of the pipeline to second state and from second state to third state of the pipeline we are using xcom (internal redis based storage offered by airflow).

After the extraction is complete, we are using postgres operator to create the following tables if they don't already exist:

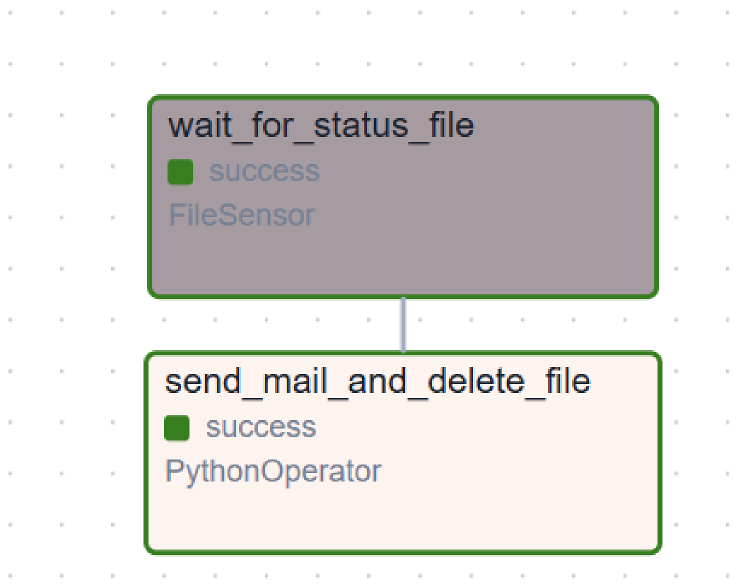
```
CREATE TABLE IF NOT EXISTS imagetable (
   "imageid" SERIAL PRIMARY KEY,
   "imagedata" BYTEA NOT NULL
);

CREATE TABLE IF NOT EXISTS storydatatable (
   "storydataid" SERIAL PRIMARY KEY,
   "imageid" integer REFERENCES imagetable (imageid),
   "headline" VARCHAR(200),
   "article_url" VARCHAR(2000) UNIQUE,
   "articledate" TIMESTAMPTZ,
   "scrapetimestamp" TIMESTAMPTZ NOT NULL DEFAULT NOW()
);
```

After creation of tables, we are fetching the extracted data in 3rd stage from xcom and inserting them into postgres database using the PostGres hook. And finally create a file named status (which contains number of successful entries into the database) in the location /opt/airflow/dags/status.

DAG 2: email_sender_and_file_deleter
Workflow:

1. Reading the file saved by the previous DAG
2. Sending an email if the previous file has a value > 0 and deleting the file



a. Wait_for_status_file: This uses a FileSensor operator which pokes in the location provided to check whether a file has been created with the specific name or not. If it finds a file then it reads it and passes the flow of control to the next operator.
b. Send_mail_and delete_file: This is a PythonOperator which has the logic for sending email if the value is > 0 and deleting the file.

Email sending is done using the library smtplib. Some non-programmatic prerequisite steps are required to set up the sender mail id account (the steps are required for generation app password which is now mandated by google, and the steps are given in the github repo).

This part takes in the sender and receiver mail id from config.py file.

Setup details:

Since I am using some extra python libraries which don't come preinstalled with python, like:

beautifulsoup4==4.12.3
opencv-python==4.11.0.86
requests

So, I have created a Dockerfile, which takes the apache/airflow:2.10.4 as the base image, and then installs these packages and spins up another image.

Dockerfile:

```
FROM apache/airflow:2.10.4
COPY requirement.txt /requirement.txt
RUN pip install --upgrade pip
RUN pip install --no-cache-dir -r /requirement.txt
```

So, first we need to build the apache/airflow:2.10.4 image from docker-compose.yaml,
Then we need to build the above mentioned Dockerfile.
Then we need to edit the docker-compose.yaml to use the image created using the above Dockerfile, and then again run docker compose up.

Also, another small change I've done, is that I have mapped another folder from host system to docker (/include), which will contain all the python scripts used from assignment-1.

I have mentioned the detailed steps in a projectSetupGuide.txt file in the github repo.