DA5402: Machine Learning Operations Laboratory

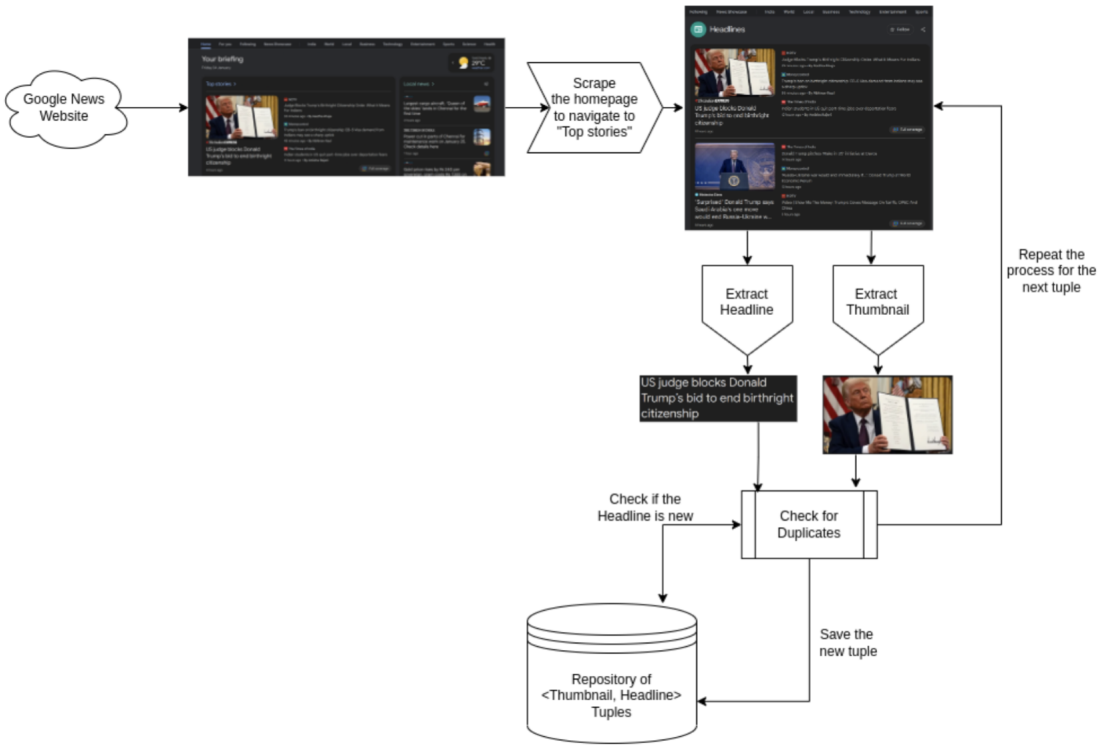
Assignment 2

Nandhakishore C S (DA24M011)

February 18, 2025

1 Problem Statement

We got introduced to Apache Airflow today, which happens to be one of most popular open-source orchestration tool amongst the industry practitioners. Let's rebuild our solution for Assignment 1 using Airflow this time. This exercise would teach you the power of using the right tool to the right problem. The workflow that we implemented in A1 is shown again here as a ready reckoner.



1.1 Module 1 [5 points]

Create a suitable Airflow Operator that would either import your Python function or directly execute your Python script to scrape the home page of Google News. The URLs to scrape may change over time, so you should keep them as configurable parameters or operator properties.

The code for module 1 is re-purposed from assignment1. This returns the HTML data from the webpage which is scrapped. The functions from airflow to pull and push data into Xcom are added.

1.2 Module 2 [5 points]

Create a suitable Airflow Operator that would either import your Python function or directly executing your Python script to scrape the "Top stories" from the home page of Google News. The URLs to scrape may change over time, so you have to keep them as configurable parameters or operator properties.

The code for module 2 is re-purposed from assignment1. This returns the URL of the Top stories section from the Google News webpage.

```python
# Module 1 - get HTML content
def module1(**kwargs):
    response = requests.get(url)
    if response.status_code == 200:
        kwargs['ti'].xcom_push(key='home_page', value=response.text)
    else:
        raise Exception(f"\nFailed to scrape:\t {url}\n")
```

Figure 1: Snap shot of the module1 from DAG1.

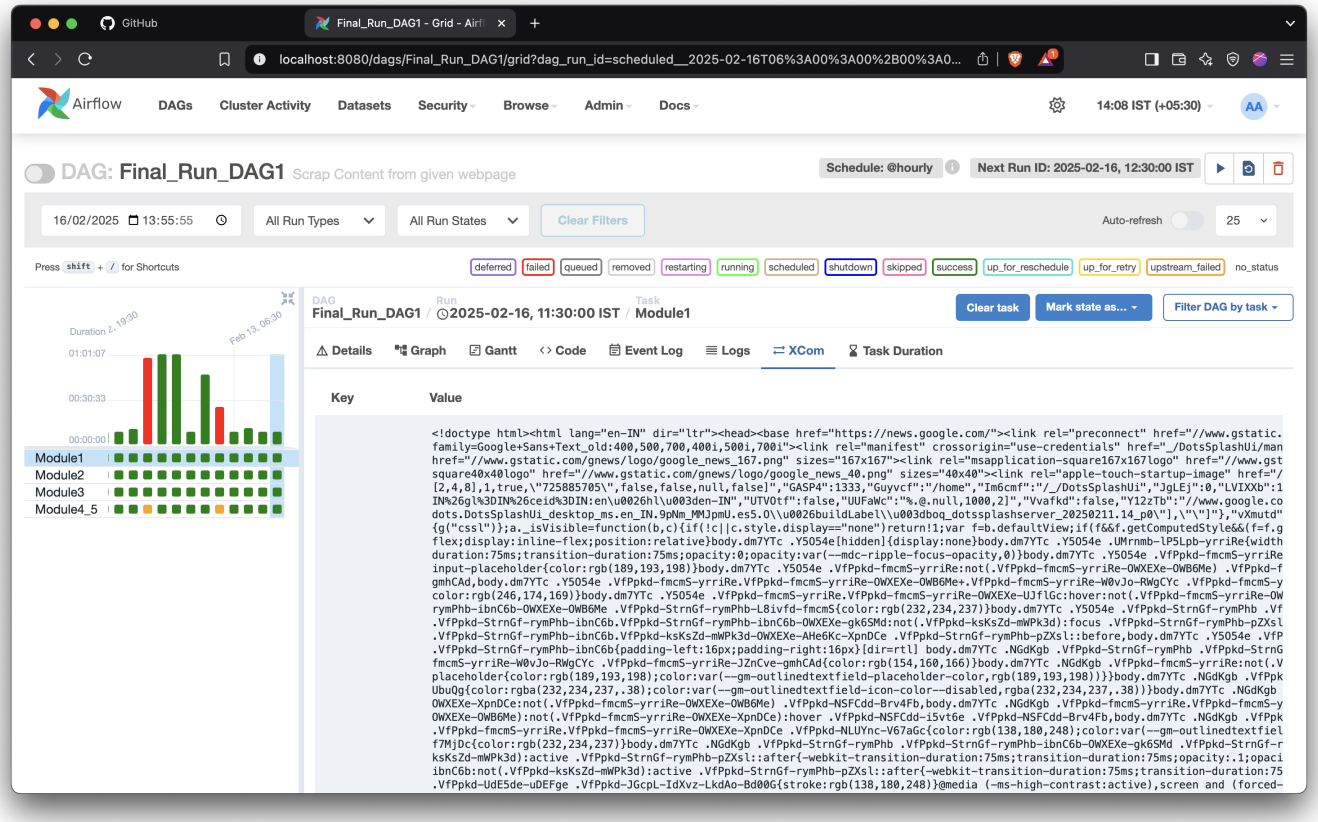


Figure 2: HTML Tags scrapped from news.google.com

```python
# Module 2 - get URL for top stories sub section
def module2(**kwargs):
    ti = kwargs['ti']
    html = ti.xcom_pull(task_ids='sub_section_url', key='home_page')
    soup = BeautifulSoup(html, 'html.parser')
    # top_story = soup.find('a', {'aria-label': 'Top stories'})
    top_story = soup.find_all('a')
    pattern = sub_section_str

    for link in top_story:
        link_text = link.get_text().strip()
        if(pattern.lower() in link_text.lower()):
            href = link.get('href')
            if(href is not None):
                # href = f'{url}{href[1 : -1]}'
                top_stories_url = f'https://news.google.com{href[1:-1]}'
                ti.xcom_push(key='top_stories_url', value=top_stories_url)
```

Figure 3: Snap shot of the module2 from DAG1.

1.3 Module 3 [10 points]

Create a suitable Airflow Operator that would either import your Python function or directly executing your Python script to extract the thumbnail and the headline of every story from that page. Remember that the page is set up for lazy loading. Your operator should factor lazy loading. The layout of the "Top stories" page may change over time, so ensure that you create your Operator in a way for easy updates later.

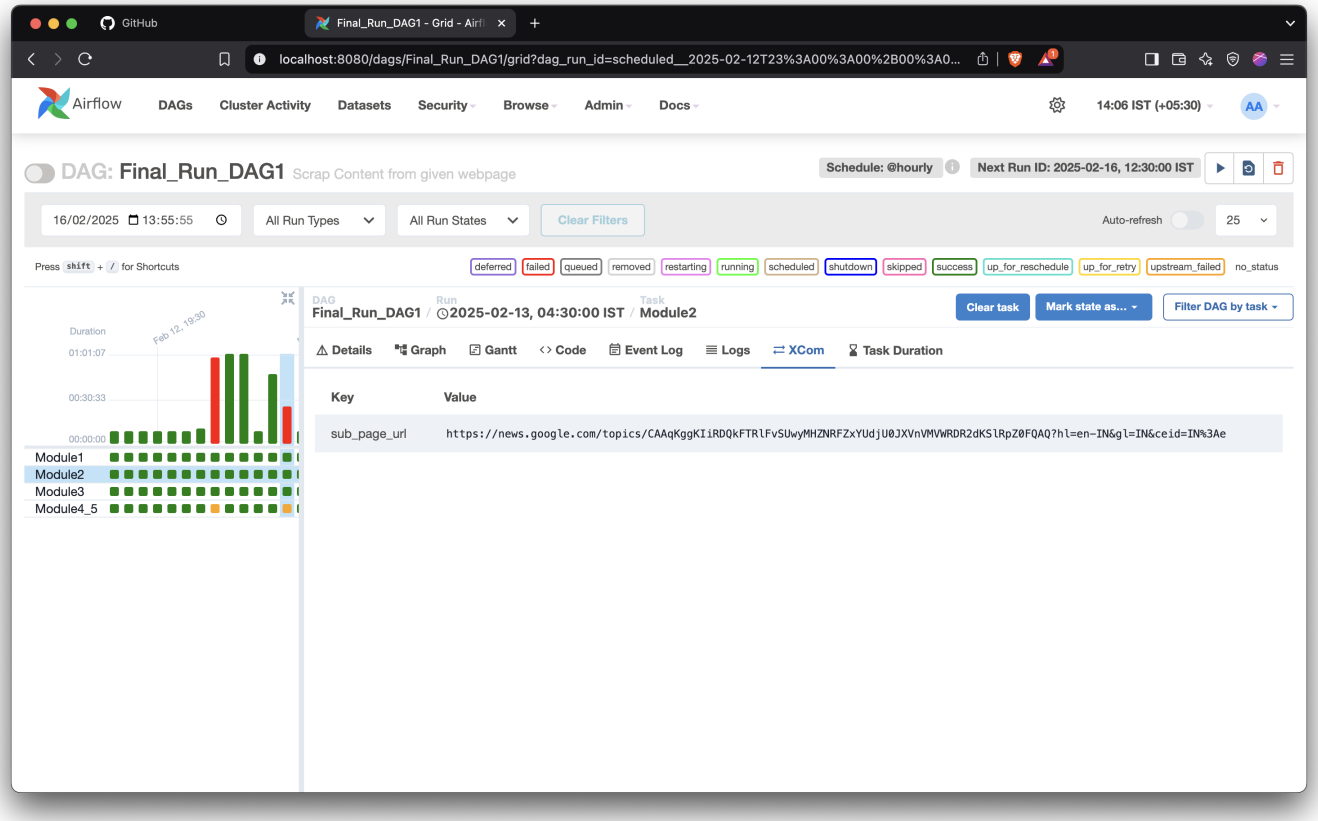


Figure 4: Top Stories URL from module2

The code for module 3 is re-purposed from assignment1. The headline, the thumbnail, date and article URL are dumped in the Airflow's Xcom as a dictionary, such that it can be accessed by a database operator if needed.

The module3 has two helper functions which are again repurposed from assignment1. The functions get the time (for getting article's publication date / save date) and to get the content (English text and URL) from the scrapped HTML tags. To handle lazy loading, a while loop with time.sleep function is utlised to wait for the page to lead. Snap shot of the module3 from DAG1.

1.4 Module 4 [10 points]

Create a Postgres operator to setup your database tables with the necessary unique/primary keys. Your database should have one table for storing the image data (base64 encoded) and the other table for storing the headlines and other meta information such as URLs, scrape timestamp, article date, etc. You can run a join query with the image and headlines tables to fetch the ¡title, image¿ pair. After the "insert" operation, you should create a status file at "dags/run/status", which should contain just one number representing the number of successful inserts. If all the records are duplicates, you will have a '0' in the status file.

1.5 Module 5 [10 points]

Setup a Airflow DAG to orchestrate the workflow pipeline with necessary properties and load them up in the Airflow console. You should setup crontab, to make it runnable every hour. As long as the Airflow container is running in the background, you will keep scrapping the articles for <image, headline> tuple

The code for module4 and module5 are combined together for the ease of coding it without errors. The function is repurposed from assignment1. The module creates a status file in the local system named 'status' and connects the DAG to a postgres database using the **psycopg2** library. This is a Airflow's Python operator which acts as a database conntector and inserts data on the database. This can viewed using the DBeaver application in the local system.
The module uses Jaccard Similarity as a measure find the similarity between the files.

```
def _jaccard_similarity(document1: str, document2: str) -> float:
    s1 = set(document1.lower().split())
```

```python
# Helper function for module3
def _extract_story_data(article: bs4.element.Tag) -> dict:
    try:
        # Extract headline and URL first
        headline = None
        for a_tag in article.find_all('a'):
            if(a_tag.get_text().strip() is not None):
                headline = a_tag.get_text().strip()
                article_url = a_tag.get('href')
                if((article_url) and (not article_url.startswith('http'))):
                    # article_url = f"https://news.google.com{article_url[1:-1]}"
                    article_url = "https://news.google.com" + article_url[1:-1]
                else:
                    break

        if(not headline):
            return None

        # Extract thumbnail
        thumbnail_url = None

        # Try figure tag first
        figure = article.find('figure')
        if(figure is not None):
            img = figure.find('img')
            if(img is not None):
                thumbnail_url = img.get('src') or img.get('data-src')
                # thumbnail_url = f"https://news.google.com{thumbnail_url}"
                thumbnail_url = "https://news.google.com" + thumbnail_url

        # If no image found, skip this story
        if(not thumbnail_url): return

        # Extract and convert publication date
        time_elem = article.find('time')
        relative_time = time_elem.get_text().strip() if time_elem else None
        pub_date = _get_time(relative_time) if relative_time else datetime.now().strftime("%Y-%m-%d %H
    :%M:%S")

        return {
            'Headline': headline,
            'Thumbnail_url': thumbnail_url,
            'Article_url': article_url,
            'Date': pub_date
        }

    except Exception as error:
        print(f"Error:\t {error}")
        return

# Helper function for module 3
def _get_time(relative_time: str) -> str:
    try:
        now = datetime.now()

        if 'minute' in relative_time or 'minutes' in relative_time:
            minutes = int(relative_time.split()[0])
            return (now - timedelta(minutes=minutes)).strftime("%Y-%m-%d %H:%M:%S")

        elif 'hour' in relative_time or 'hours' in relative_time:
            hours = int(relative_time.split()[0])
            return (now - timedelta(hours=hours)).strftime("%Y-%m-%d %H:%M:%S")

        elif 'day' in relative_time or 'days' in relative_time:
            days = int(relative_time.split()[0])
            return (now - timedelta(days=days)).strftime("%Y-%m-%d %H:%M:%S")

        elif 'Yesterday' in relative_time:
            return (now - timedelta(days=1)).strftime("%Y-%m-%d %H:%M:%S")

        else:
            return now.strftime("%Y-%m-%d %H:%M:%S")

    except Exception:
        return now.strftime("%Y-%m-%d %H:%M:%S")
```

Figure 5: Helpers functions for module3 from DAG1.

```python
# Module 3 - get thumbnail, headlines and META data
def module3(**kwargs):
    ti = kwargs['ti']
    top_stories_url = ti.xcom_pull(task_ids='extract_top_stories', key='top_stories_url')

    stories = []
    page = 1
    while(page <= 5):
        try:
            response = requests.get(top_stories_url)
            response.raise_for_status()
            soup = BeautifulSoup(response.text, 'html.parser')
            articles = soup.find_all('article')

            for article in articles:
                story = _extract_story_data(article)
                if story and story not in stories:
                    stories.append(story)

            if ((not articles) or (len(articles) == 0)):
                break

            page += 1
            time.sleep(5)

        except requests.RequestException as error:
            print(f'Error fetching {page}: {error}')
            break

    ti.xcom_push(key='news_data', value=stories)
```

Figure 6: Snap shot of the module3 from DAG1.

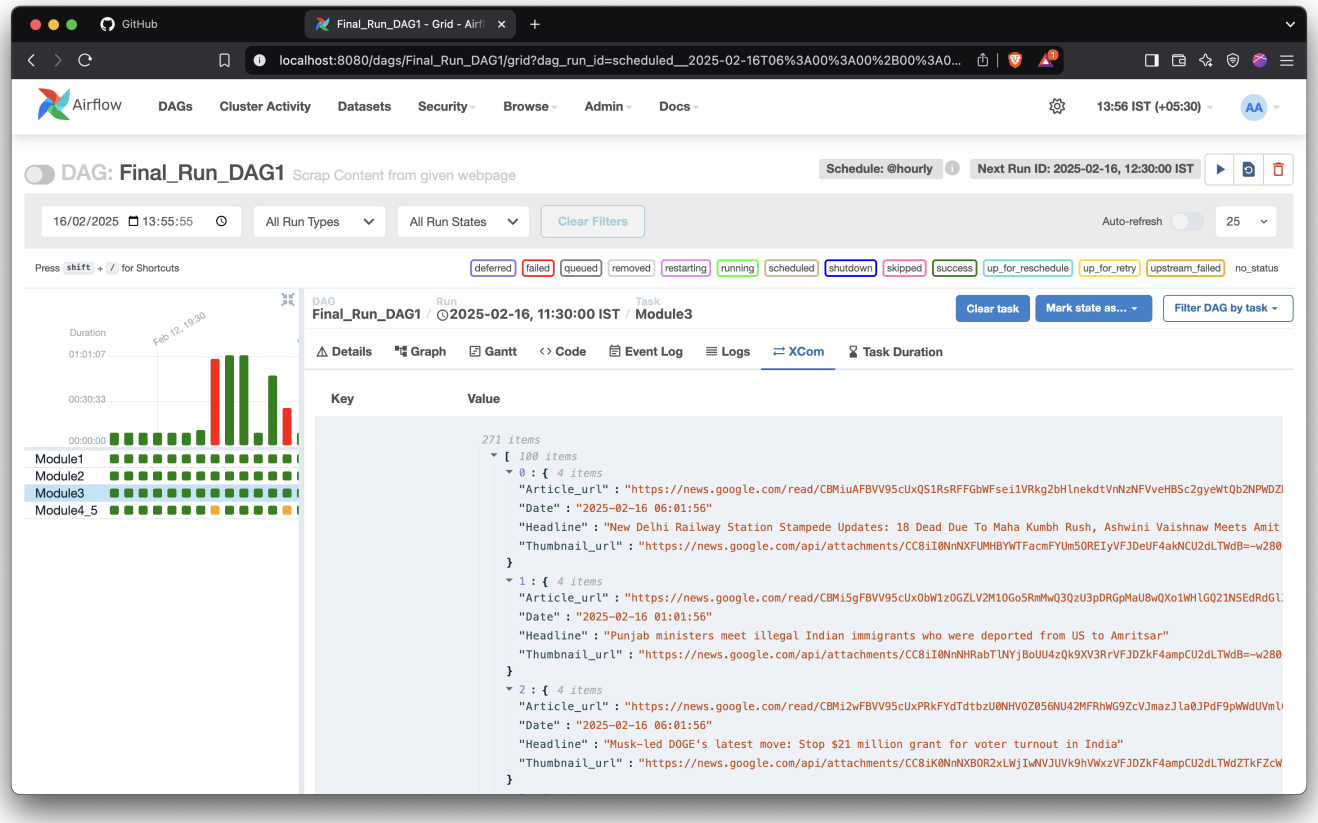


Figure 7: Data scrapped from top stories page

```python
    s2 = set(document2.lower().split())

    intersection = len(s1.intersection(s2))
    union = len(s1.union(s2))

    return intersection/ union if union > 0 else 0
```

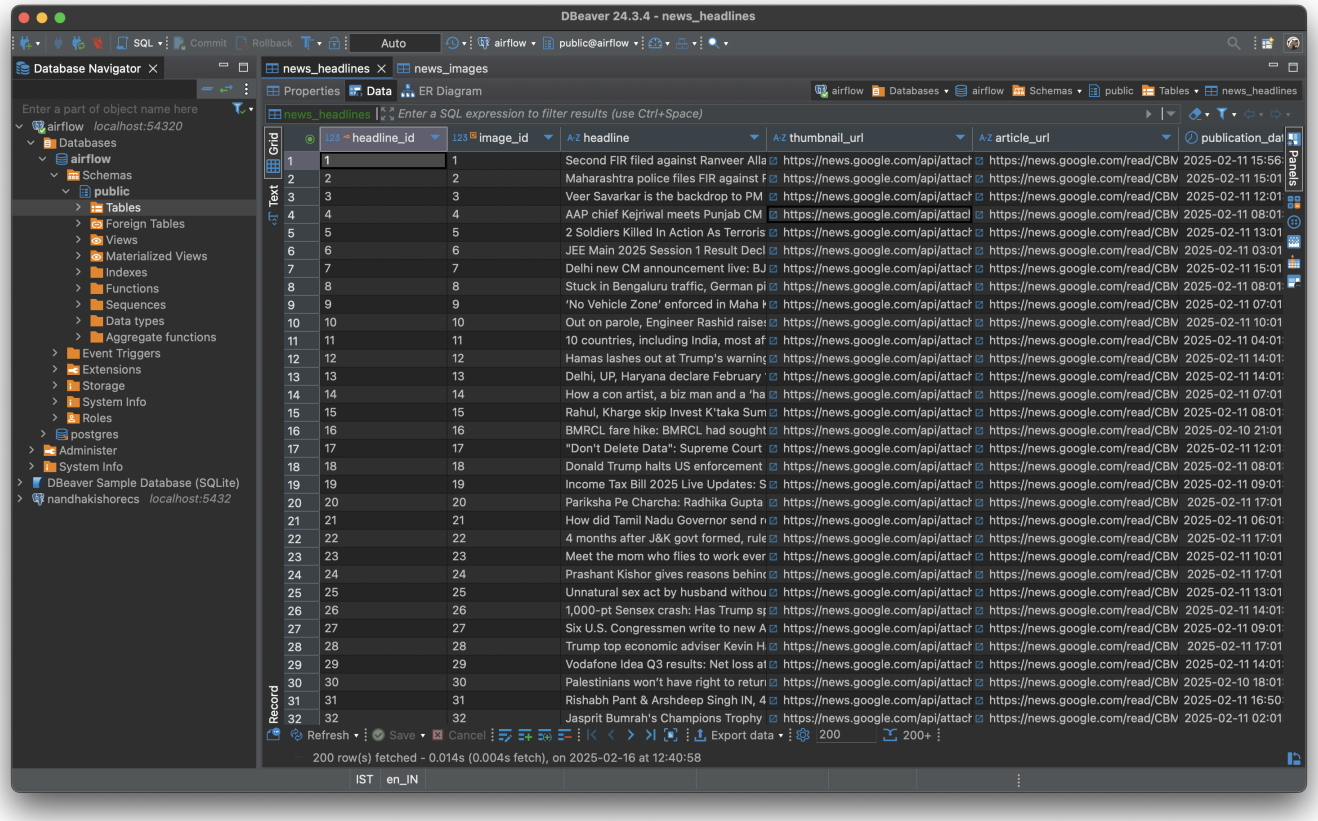


Figure 8: Table in database with entries of headlines, article URL , thumbnail URL and meta data. The column headline ID is used as a foreign key to infer the other table to save the thumbnail image

```
def is_similar(new_headline, existing_headlines):
    for headline in existing_headlines:
        if SequenceMatcher(None, new_headline, headline).ratio() > 0.85:
            return True
    return False
```

Helper function to check similarity between headlines for module 4 and 5

```
def module4(**kwargs):
    try:
        ti = kwargs['ti']
        news_data = ti.xcom_pull(task_ids='extract_news_data', key='news_data')
        pg_hook = PostgresHook(postgres_conn_id = postgres_connection_ID)
        connection = pg_hook.get_conn()
        cursor = connection.cursor()

        # counter variable for successfull inserts
        counter = 0

        for story in news_data:
            cursor.execute(
                "SELECT Headline FROM news_headlines"
            )
            current_headlines = cursor.fetchall()

            duplicate = False
            for (current_headline, ) in current_headlines:
                # similarity = is_similar(story['Headline'], current_headline)
                # if(similarity is not True):
                #     print("\nDuplicate values of headline found!\n")
                #     duplicate = True
                #     break
                similarity = _jaccard_similarity(story['Headline'], current_headline)
                if(similarity >= .75):
                    print("\nDuplicate values of headline found!\n")
                    duplicate = True
                    break
```

```
            if(duplicate == True):
                continue

            thumbnail_url = story['Thumbnail_url']
            thumbnail_data = requests.get(thumbnail_url)
            thumbnail_data.raise_for_status()
            thumbnail_image = base64.b64encode(thumbnail_data.content).decode('utf-8')

            cursor.execute(
                "INSERT INTO news_images (image_data) VALUES (%s) RETURNING image_id",
                (thumbnail_image,)
            )

            thumbnail_ID = cursor.fetchone()[0]

            cursor.execute(
                """
                    INSERT INTO news_headlines
                    (image_id, headline, thumbnail_url, article_url, publication_date,
    scrape_timestamp)
                    VALUES (%s, %s, %s, %s, %s, %s)
                    ON CONFLICT (headline, article_url) DO NOTHING
                    RETURNING headline_id
                """,
                (
                    thumbnail_ID,
                    story['Headline'],
                    story['Thumbnail_url'],
                    story['Article_url'],
                    story['Date'],
                    datetime.now()
                )
            )

            if (cursor.fetchone() is not None):
                counter += 1
            else:
                cursor.execute(
                    "DELETE FROM news_images WHERE image_id = %s", (thumbnail_ID, )
                )
        # connection commit
        connection.commit()

        # Status check
        dir_path = '/opt/airflow/dags'
        os.makedirs(dir_path, exist_ok = True)

        with open(os.path.join(dir_path, 'status'), 'w') as file:
            file.write(str(counter))

    except Exception as error:
        print(error)
        raise
```

1.6 Module 6[10 points]

Setup another Airflow DAG that would send an email to yourself, whenever a new ¡image, headline¿ tuple is added to the database. This DAG has to remember the previous state of the database tables to detect new rows. When news rows are detected, it should it's knowledge of the previous state and also send an email to your inbox. To send emails, you may have to learn about SMTP + TLS/SSL setup with credentials. The set up is the same process when you configure your email client (Thunderbird, Outlook, etc) to send emails from your gmail or institute accounts. The only difference is you are going to do it programmatically.

The module to check for new entries and send email requires the two factor authentication code from the Gmail account. Insert your code and email details at the start of the DAG.

```
# Configuration to send notification email using DAGs
SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 587
SENDER_EMAIL = "" # sender email
PASSWORD = "" # send 2FA pass key
RECIPIENT_EMAIL = "" # reciepient email

# ------------------------------ Helper Function to do Module 6 --------------------------------
def _send_email(message):
    # Send email using SMTP with App Password
    try:
        with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:
```

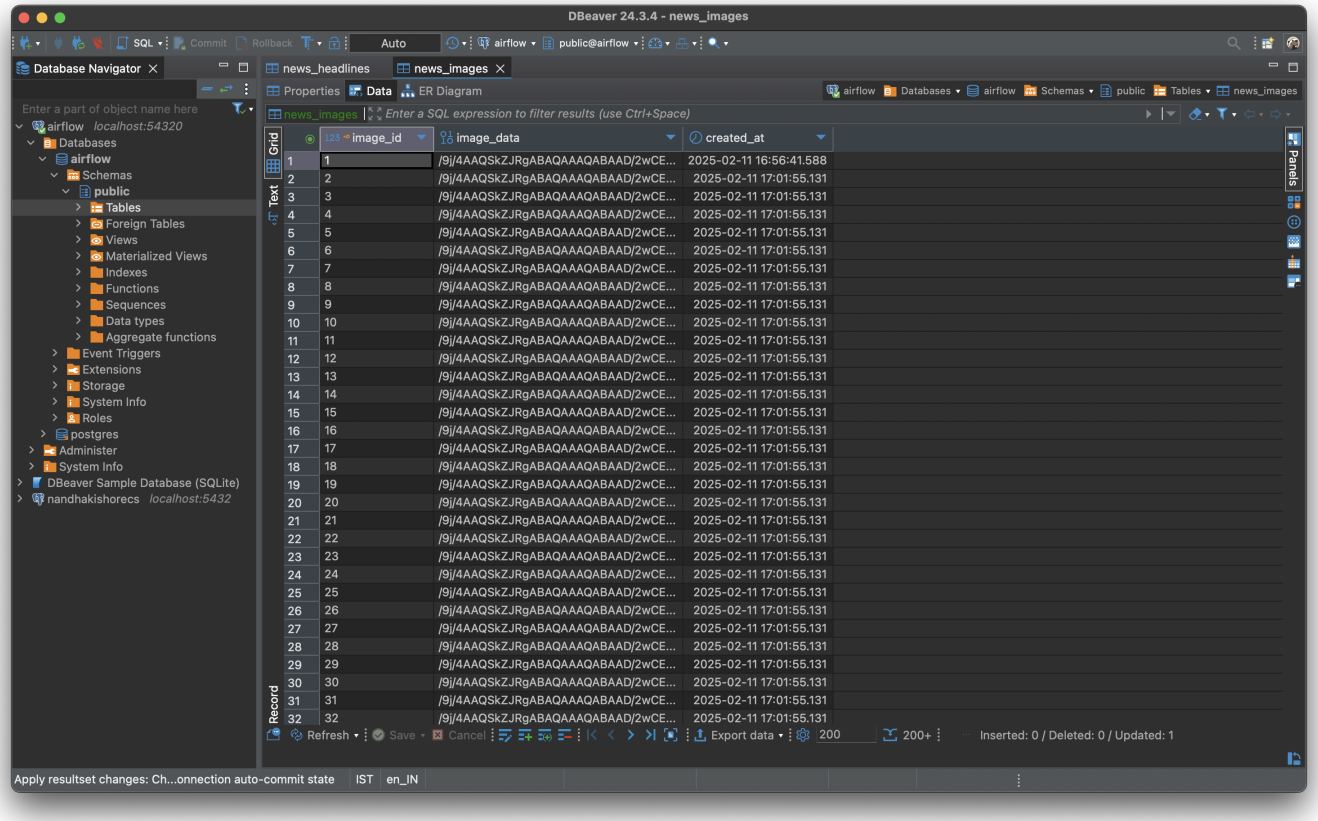


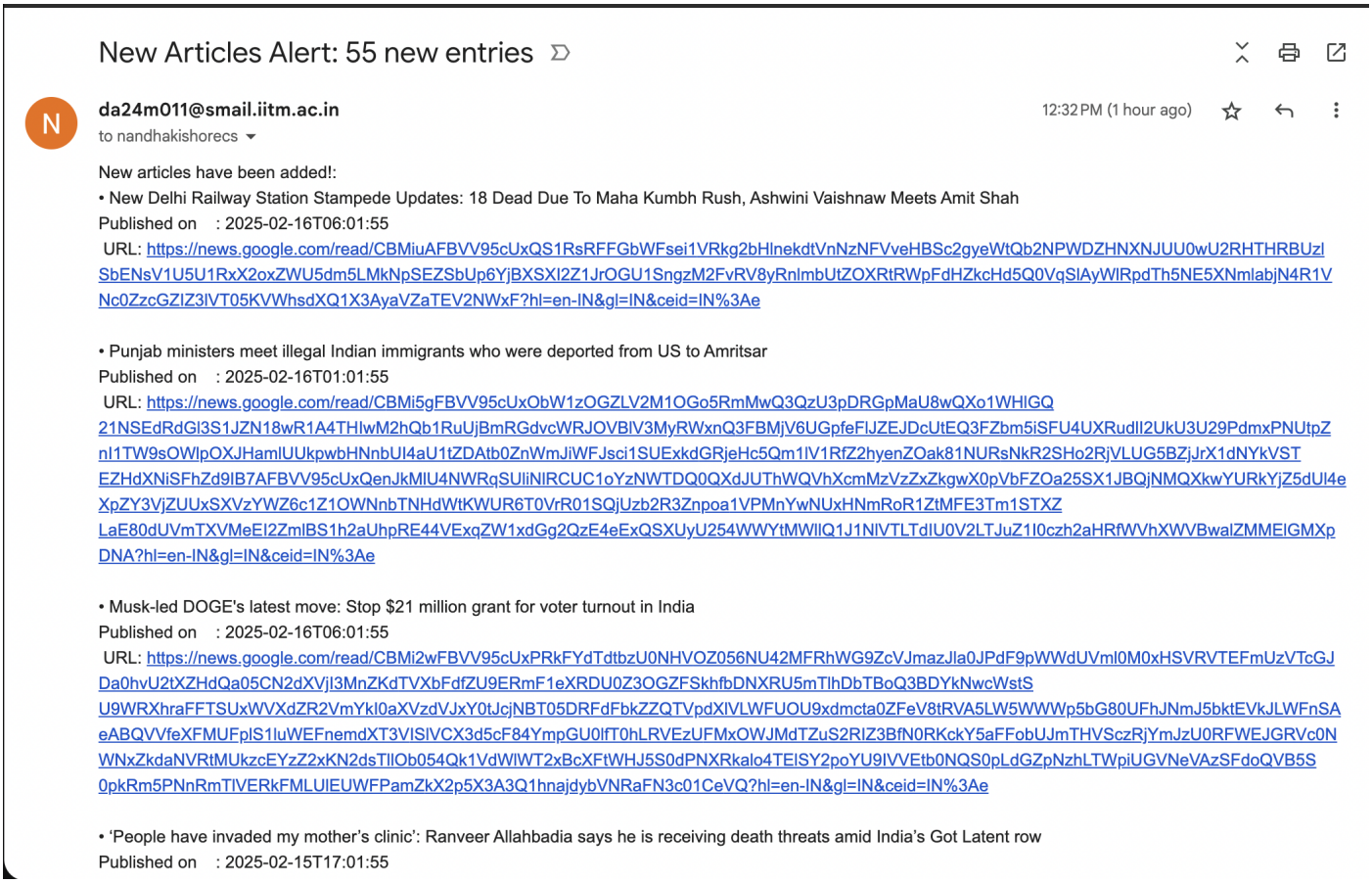
Figure 9: Table with thumbnail data in base64 encoded form

```
            server.starttls()
            # Using App Password here - get token from GMail 2FA
            server.login(SENDER_EMAIL, PASSWORD)
            server.send_message(message)
            print("Email sent successfully")
    except Exception as error:
        print(f"Failed to send email: {error}")
        raise

# ------------------------------- Helper Function to do Module 6 --------------------------------
def _load_previous_state():
    # Load previous database content - the pnultimate state from a JSON file
    try:
        with open('/opt/airflow/dags/previous_state.json', 'r') as file:
            return json.load(file)
    except FileNotFoundError:
        return {'last_headline_id': 0}

# ------------------------------- Helper Function to do Module 6 --------------------------------
def save_current_state(state):
    # Save current database state to a JSON file
    with open('/opt/airflow/dags/previous_state.json', 'w') as file:
        json.dump(state, file)

# ------------------------------- Helper Function to do Module 6 --------------------------------
def _compose_email(new_entries):
    # To email the new changes compose an email with content from new entries
    message = MIMEMultipart()
    message["Subject"] = f"New Articles Alert: {len(new_entries)} new entries"
    message["From"] = SENDER_EMAIL
    message["To"] = RECIPIENT_EMAIL

    if(len(new_entries) > 0):
        content = "New articles have been added!:\n"
        for entry in new_entries:
            content += f"    {entry['headline']}\n"
            if entry.get('publication_date'):
                content += f"Published on\t: {entry['publication_date']}\n"
            content += f" URL: {entry['article_url']}\n\n"

        message.attach(MIMEText(content, "plain"))
```

```
    else:
        content = 'No new top stories!\n'
        message.attach(MIMEText(content, "plain"))
    return message

# ------------------------------ Main Function for Module 6 ------------------------------
def module6(**context):
    try:
        # Get previous state
        prev_state = _load_previous_state()
        last_headline_id = prev_state['last_headline_id']

        # Connect to database
        pg_hook = PostgresHook(postgres_conn_id = postgres_connection_ID)
        connection = pg_hook.get_conn()
        cursor = connection.cursor()

        # Get new entries - SQL Query
        cursor.execute(
            """
                SELECT h.headline_id, h.Headline, h.Article_url, h.publication_date
                FROM news_headlines h
                WHERE h.headline_id > %s
                ORDER BY h.headline_id
            """,
            (last_headline_id, )
        )

        new_entries = []
        max_headline_id = last_headline_id

        for row in cursor.fetchall():
            headline_id, headline, article_url, pub_date = row
            max_headline_id = max(max_headline_id, headline_id)
            new_entries.append({
                'headline': headline,
                'article_url': article_url,
                'publication_date': pub_date.isoformat() if pub_date else None
            })

        if (new_entries is not None):
            # Prepare and send email
            message = _compose_email(new_entries)
            _send_email(message)

            # Update state
            save_current_state({'last_headline_id': max_headline_id})

        # Clean up status file
        os.remove('/opt/airflow/dags/status')

    except Exception as error:
        print(f"Error in sending email: {error}")
        raise

    finally:
        if 'cursor' in locals():
            cursor.close()
        if 'connection' in locals():
            connection.close()
```

Function for module6 from DAG2.

```
# Configuration to send notification email using DAGs
SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 587
SENDER_EMAIL = "" # sender email
PASSWORD = "" # send 2FA pass key
RECIPIENT_EMAIL = "" # reciepient email

# ------------------------------ Helper Function to do Module 6 ------------------------------
def _send_email(message):
    # Send email using SMTP with App Password
    try:
        with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:
            server.starttls()
            # Using App Password here - get token from GMail 2FA
            server.login(SENDER_EMAIL, PASSWORD)
            server.send_message(message)
            print("Email sent successfully")
```



New Articles Alert: 55 new entries

N

da24m011@smail.iitm.ac.in 12:32 PM (1 hour ago)

to nandhakishorecs

New articles have been added!:

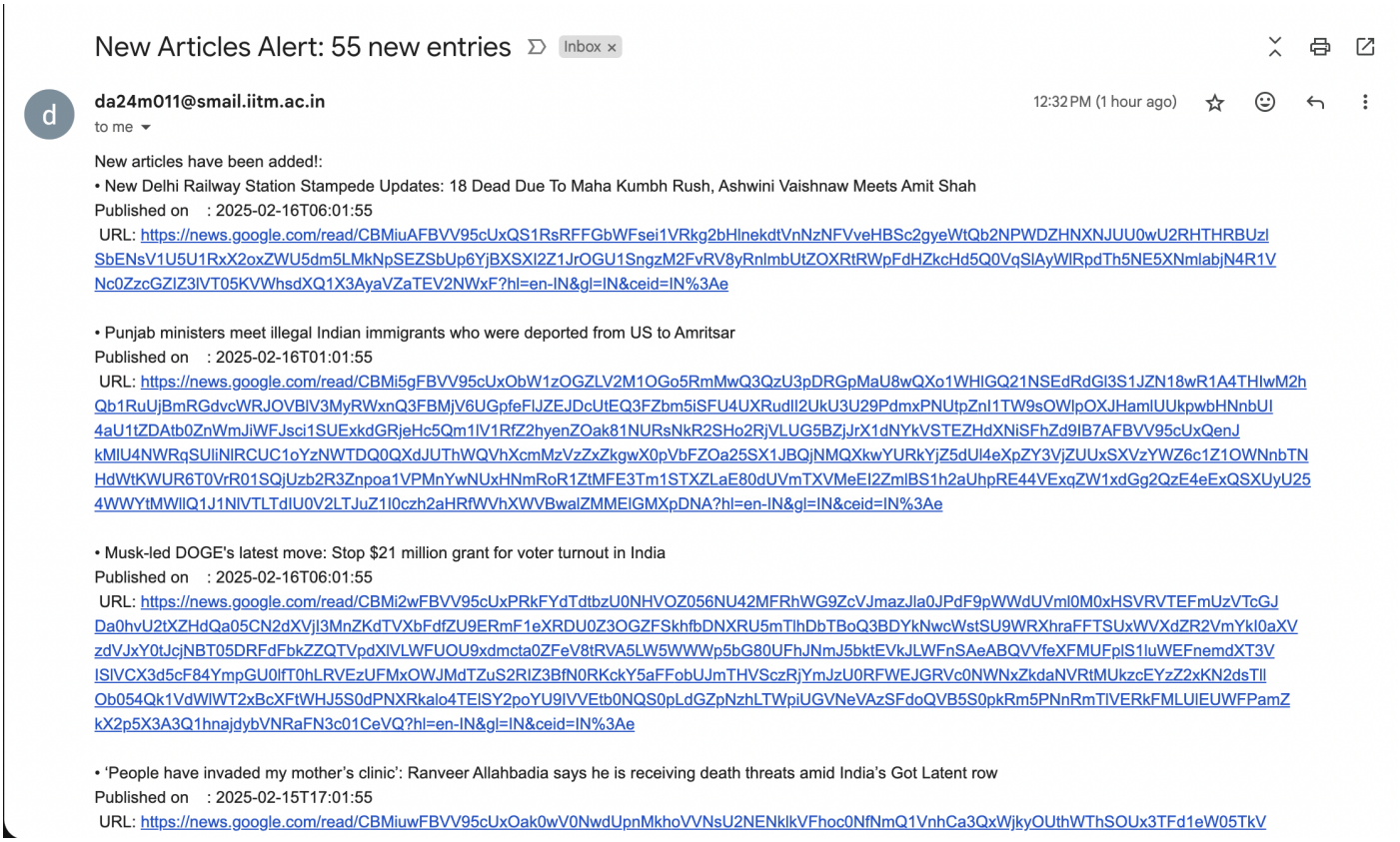
• New Delhi Railway Station Stampede Updates: 18 Dead Due To Maha Kumbh Rush, Ashwini Vaishnaw Meets Amit Shah
Published on : 2025-02-16T06:01:55
URL: https://news.google.com/read/CBMiuAFBVV95cUxQS1RsRFFGbWFsei1VRkg2bHlnekdtVnNzNFVveHBSc2gyeWtQb2NPWDZHNXNJUU0wU2RHTHRBUzlSbENsV1U5U1RxX2oxZWU5dm5LMkNpSEZSbUp6YjBXSXI2Z1JrOGU1SngzM2FvRV8yRnlmbUtZOXRtRWpFdHZkcHd5Q0VqSlAyWlRpdTh5NE5XNmlabjN4R1VNc0ZzcGZlZ3lVT05KVWhsdXQ1X3AyaVZaTEV2NWxF?hl=en-IN&gl=IN&ceid=IN%3Ae

• Punjab ministers meet illegal Indian immigrants who were deported from US to Amritsar
Published on : 2025-02-16T01:01:55
URL: https://news.google.com/read/CBMi5gFBVV95cUxObW1zOGZLV2M1OGo5RmMwQ3QzU3pDRGpMaU8wQXo1WHlGQ21NSEdRdGl3S1JZN18wR1A4THIwM2hQb1RuUjBmRGdvcWRJOVBlV3MyRWxnQ3FBMjV6UGpfeFlJZEJDcUtEQ3FZbm5iSFU4UXRudlI2UkU3U29PdmxPNUtpZnI1TW9sOWlpOXJHamlUUkpwbHNnbUl4aU1tZDAtb0ZnWmJiWFJsci1SUExkdGRjeHc5Qm1lV1RfZ2hyenZOak81NURsNkR2SHo2RjVLUG5BZjJrX1dNYkVSTEZHdXNiSFhZd9lB7AFBVV95cUxQenJkMlU4NWRqSUliNlRCUC1oYzNWTDQ0QXdJUThWQVhXcmMzVzZxZkgwX0pVbFZOa25SX1JBQjNMQXkwYURkYjZ5dUl4eXpZY3VjZUUxSXVzYWZ6c1Z1OWNnbTNHdWtKWUR6T0VrR01SQjUzb2R3Znpoa1VPMnYwNUxHNmRoR1ZtMFE3Tm1STXZLaE80dUVmTXVMeEI2ZmlBS1h2aUhpRE44VExqZW1xdGg2QzE4eExQSXUyU254WWYtMWllQ1J1NlVTLTdIU0V2LTJuZ1I0czh2aHRfWVhXWVBwalZMMElGMXpDNA?hl=en-IN&gl=IN&ceid=IN%3Ae

• Musk-led DOGE's latest move: Stop $21 million grant for voter turnout in India
Published on : 2025-02-16T06:01:55
URL: https://news.google.com/read/CBMi2wFBVV95cUxPRkFYdTdtbzU0NHVOZ056NU42MFRhWG9ZcVJmazJla0JPdF9pWWdUVml0M0xHSVRVTEFmUzVTcGJDa0hvU2tXZHdQa05CN2dXVjI3MnZKdTVXbFdfZU9ERmF1eXRDU0Z3OGZFSkhfbDNXRU5mTlhDbTBoQ3BDYkNwcWstSU9WRXhraFFTSUxWVXdZR2VmYkI0aXVzdVJxY0tJcjNBT05DRFdFbkZZQTVpdXlVLWFUOU9xdmcta0ZFeV8tRVA5LW5WWWp5bG80UFhJNmJ5bktEVkJLWFnSAeABQVVfeXFMUFplS1luWEFnemdXT3VISlVCX3d5cF84YmpGU0lfT0hLRVEzUFMxOWJMdTZuS2RIZ3BfN0RKckY5aFFobUJmTHVSczRjYmJzU0RFWEJGRVc0NWNxZkdaNVRtMUkzcEYzZ2xKN2dsTllOb054Qk1VdWlWT2xBcXFtWHJ5S0dPNXRkalo4TElSY2poYU9lVVEtb0NQS0pLdGZpNzhLTWpiUGVNeVAzSFdoQVB5S0pkRm5PNnRmTlVERkFMLUlEUWFPamZkX2p5X3A3Q1hnajdybVNRaFN3c01CeVQ?hl=en-IN&gl=IN&ceid=IN%3Ae

• 'People have invaded my mother's clinic': Ranveer Allahbadia says he is receiving death threats amid India's Got Latent row
Published on : 2025-02-15T17:01:55

Figure 10: Emails being send from DAG2 to about new additions of news articles



New Articles Alert: 55 new entries Inbox ×

d

da24m011@smail.iitm.ac.in 12:32 PM (1 hour ago)

to me

New articles have been added!:

• New Delhi Railway Station Stampede Updates: 18 Dead Due To Maha Kumbh Rush, Ashwini Vaishnaw Meets Amit Shah
Published on : 2025-02-16T06:01:55
URL: https://news.google.com/read/CBMiuAFBVV95cUxQS1RsRFFGbWFsei1VRkg2bHlnekdtVnNzNFVveHBSc2gyeWtQb2NPWDZHNXNJUU0wU2RHTHRBUzlSbENsV1U5U1RxX2oxZWU5dm5LMkNpSEZSbUp6YjBXSXI2Z1JrOGU1SngzM2FvRV8yRnlmbUtZOXRtRWpFdHZkcHd5Q0VqSlAyWlRpdTh5NE5XNmlabjN4R1VNc0ZzcGZlZ3lVT05KVWhsdXQ1X3AyaVZaTEV2NWxF?hl=en-IN&gl=IN&ceid=IN%3Ae

• Punjab ministers meet illegal Indian immigrants who were deported from US to Amritsar
Published on : 2025-02-16T01:01:55
URL: https://news.google.com/read/CBMi5gFBVV95cUxObW1zOGZLV2M1OGo5RmMwQ3QzU3pDRGpMaU8wQXo1WHlGQ21NSEdRdGl3S1JZN18wR1A4THIwM2hQb1RuUjBmRGdvcWRJOVBlV3MyRWxnQ3FBMjV6UGpfeFlJZEJDcUtEQ3FZbm5iSFU4UXRudlI2UkU3U29PdmxPNUtpZnI1TW9sOWlpOXJHamlUUkpwbHNnbUl4aU1tZDAtb0ZnWmJiWFJsci1SUExkdGRjeHc5Qm1lV1RfZ2hyenZOak81NURsNkR2SHo2RjVLUG5BZjJrX1dNYkVSTEZHdXNiSFhZd9lB7AFBVV95cUxQenJkMlU4NWRqSUliNlRCUC1oYzNWTDQ0QXdJUThWQVhXcmMzVzZxZkgwX0pVbFZOa25SX1JBQjNMQXkwYURkYjZ5dUl4eXpZY3VjZUUxSXVzYWZ6c1Z1OWNnbTNHdWtKWUR6T0VrR01SQjUzb2R3Znpoa1VPMnYwNUxHNmRoR1ZtMFE3Tm1STXZLaE80dUVmTXVMeEI2ZmlBS1h2aUhpRE44VExqZW1xdGg2QzE4eExQSXUyU254WWYtMWllQ1J1NlVTLTdIU0V2LTJuZ1I0czh2aHRfWVhXWVBwalZMMElGMXpDNA?hl=en-IN&gl=IN&ceid=IN%3Ae

• Musk-led DOGE's latest move: Stop $21 million grant for voter turnout in India
Published on : 2025-02-16T06:01:55
URL: https://news.google.com/read/CBMi2wFBVV95cUxPRkFYdTdtbzU0NHVOZ056NU42MFRhWG9ZcVJmazJla0JPdF9pWWdUVml0M0xHSVRVTEFmUzVTcGJDa0hvU2tXZHdQa05CN2dXVjI3MnZKdTVXbFdfZU9ERmF1eXRDU0Z3OGZFSkhfbDNXRU5mTlhDbTBoQ3BDYkNwcWstSU9WRXhraFFTSUxWVXdZR2VmYkI0aXVzdVJxY0tJcjNBT05DRFdFbkZZQTVpdXlVLWFUOU9xdmcta0ZFeV8tRVA5LW5WWWp5bG80UFhJNmJ5bktEVkJLWFnSAeABQVVfeXFMUFplS1luWEFnemdXT3VISlVCX3d5cF84YmpGU0lfT0hLRVEzUFMxOWJMdTZuS2RIZ3BfN0RKckY5aFFobUJmTHVSczRjYmJzU0RFWEJGRVc0NWNxZkdaNVRtMUkzcEYzZ2xKN2dsTllOb054Qk1VdWlWT2xBcXFtWHJ5S0dPNXRkalo4TElSY2poYU9lVVEtb0NQS0pLdGZpNzhLTWpiUGVNeVAzSFdoQVB5S0pkRm5PNnRmTlVERkFMLUlEUWFPamZkX2p5X3A3Q1hnajdybVNRaFN3c01CeVQ?hl=en-IN&gl=IN&ceid=IN%3Ae

• 'People have invaded my mother's clinic': Ranveer Allahbadia says he is receiving death threats amid India's Got Latent row
Published on : 2025-02-15T17:01:55
URL: https://news.google.com/read/CBMiuwFBVV95cUxOak0wV0NwdUpnMkhoVVNsU2NENklkVFhoc0NfNmQ1VnhCa3QxWjkyOUthWThSOUx3TFd1eW05TkV

Figure 11: Emails received from DAG2 about new headlines

```
    except Exception as error:
        print(f"Failed to send email: {error}")
        raise

# ------------------------------- Helper Function to do Module 6 -------------------------------
def _load_previous_state():
    # Load previous database content - the pnultimate state from a JSON file
    try:
        with open('/opt/airflow/dags/previous_state.json', 'r') as file:
            return json.load(file)
    except FileNotFoundError:
        return {'last_headline_id': 0}

# ------------------------------- Helper Function to do Module 6 -------------------------------
def save_current_state(state):
    # Save current database state to a JSON file
    with open('/opt/airflow/dags/previous_state.json', 'w') as file:
        json.dump(state, file)

# ------------------------------- Helper Function to do Module 6 -------------------------------
def _compose_email(new_entries):
    # To email the new changes compose an email with content from new entries
    message = MIMEMultipart()
    message["Subject"] = f"New Articles Alert: {len(new_entries)} new entries"
    message["From"] = SENDER_EMAIL
    message["To"] = RECIPIENT_EMAIL

    if(len(new_entries) > 0):
        content = "New articles have been added!:\n"
        for entry in new_entries:
            content += f"    {entry['headline']}\n"
            if entry.get('publication_date'):
                content += f"Published on\t: {entry['publication_date']}\n"
            content += f" URL: {entry['article_url']}\n\n"

        message.attach(MIMEText(content, "plain"))
    else:
        content = 'No new top stories!\n'
        message.attach(MIMEText(content, "plain"))
    return message

# ------------------------------- Main Function for Module 6 -------------------------------
def module6(**context):
    try:
        # Get previous state
        prev_state = _load_previous_state()
        last_headline_id = prev_state['last_headline_id']

        # Connect to database
        pg_hook = PostgresHook(postgres_conn_id = postgres_connection_ID)
        connection = pg_hook.get_conn()
        cursor = connection.cursor()

        # Get new entries - SQL Query
        cursor.execute(
            """
                SELECT h.headline_id, h.Headline, h.Article_url, h.publication_date
                FROM news_headlines h
                WHERE h.headline_id > %s
                ORDER BY h.headline_id
            """,
            (last_headline_id, )
        )

        new_entries = []
        max_headline_id = last_headline_id

        for row in cursor.fetchall():
            headline_id, headline, article_url, pub_date = row
            max_headline_id = max(max_headline_id, headline_id)
            new_entries.append({
                'headline': headline,
                'article_url': article_url,
                'publication_date': pub_date.isoformat() if pub_date else None
            })

        if (new_entries is not None):
            # Prepare and send email
            message = _compose_email(new_entries)
            _send_email(message)

```

```
            # Update state
            save_current_state({'last_headline_id': max_headline_id})

        # Clean up status file
        os.remove('/opt/airflow/dags/status')

    except Exception as error:
        print(f"Error in sending email: {error}")
        raise

    finally:
        if 'cursor' in locals():
            cursor.close()
        if 'connection' in locals():
            connection.close()
```

```
# ------------------------------ Define default_args for DAGs ------------------------------
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2024, 2, 18),
    'retries': 2,
    'retry_delay': timedelta(minutes=5),
}

# ------------------------------ DAG Configuration ------------------------------
with DAG('DAG_1_FINAL', default_args=default_args, schedule_interval='@hourly', catchup=False) as dag1:
    scrape_task = PythonOperator(task_id='sub_section_url', python_callable=module1, provide_context=True)
    extract_task = PythonOperator(task_id='extract_top_stories', python_callable=module2, provide_context=True)
    get_news_task = PythonOperator(task_id='extract_news_data', python_callable=module3, provide_context=True)
    create_table_task = PostgresOperator(
        task_id = 'create_tables',
        postgres_conn_id = postgres_connection_ID,
        sql= """
            DO $$
            BEGIN
                -- Create news_images table if it doesn't exist
                IF NOT EXISTS (SELECT FROM pg_tables WHERE schemaname = 'public' AND tablename = 'news_images') THEN
                    CREATE TABLE news_images (
                        image_id SERIAL PRIMARY KEY,
                        image_data BYTEA NOT NULL,
                        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
                    );
                END IF;

                -- Create news_headlines table if it doesn't exist
                IF NOT EXISTS (SELECT FROM pg_tables WHERE schemaname = 'public' AND tablename = 'news_headlines') THEN
                    CREATE TABLE news_headlines (
                        headline_id SERIAL PRIMARY KEY,
                        image_id INTEGER REFERENCES news_images(image_id),
                        headline TEXT NOT NULL,
                        thumbnail_url TEXT,
                        article_url TEXT,
                        publication_date TIMESTAMP,
                        scrape_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
                        UNIQUE(headline, article_url)
                    );
                END IF;
            END $$;
        """
    )
    store_data_task = PythonOperator(task_id='store_news_data', python_callable=module4, provide_context=True)
    create_table_task >> scrape_task >> extract_task >> get_news_task >> store_data_task


with DAG('DAG_2_FINAL', default_args=default_args, schedule_interval='@hourly', catchup=False) as dag2:
    wait_for_dag1 = ExternalTaskSensor(
        task_id = 'wait_for_scraping',
        # The external tag ID should be same as DAG1
        external_dag_id = 'DAG_1_FINAL',
        # Wait for the last task of DAG1
        external_task_id = 'store_news_data',
        timeout = 600,
```

Figure 12: DAG 1 and 2 with arguments taken from the user

```
        poke_interval = 60
    )

    # Wait for status file
    wait_for_status = FileSensor(
        task_id = 'wait_for_status_file',
        # Use 'file' as connection ID - cope this from DAG Dashboard - admin - connections
        fs_conn_id = postgres_connection_ID,
        # For macbook, the opt arguement is required.
        filepath = '/opt/airflow/dags/status',
        poke_interval = 30,
        timeout = 600
    )

    # Process new entries and send email
    notify_task = PythonOperator(
        task_id = 'Module6',
        python_callable = module6,
        provide_context = True
    )

    wait_for_dag1 >> wait_for_status >> notify_task
```