

Cloud Analytics and Data Warehouse Implementation - NYT Dataset

Introduction

In an era marked by exponential data growth and evolving business landscapes, the ability to derive actionable insights from vast datasets is paramount. This report encapsulates our collaborative efforts in conceptualizing, designing, and executing a comprehensive solution that integrates historical data analysis, real-time streaming, and business intelligence. Guided by the project guidelines, we navigated through intricate stages, from dataset selection to the deployment of advanced analytics techniques.

Dataset Selection and Importance

The selection of the New York Times Headlines dataset encompasses both archived data and real-time streaming API data. This Archive dataset contains a comprehensive collection of articles from The New York Times, spanning from January 1, 2000, to almost August 1, 2000 with the first 100mb of the dataset available in [Kaggle](#). For the realtime component, [ArticleAPI](#) and [ArchiveAPI](#) are chosen to fetch recent articles to be added to the already present archive data. We will explore the detailed steps as we proceed.

Database Configuration

For our database configuration, we've chosen MySQL hosted on Amazon RDS. This combination offers reliability and scalability, crucial for managing our data effectively. MySQL's robust feature set ensures our database meets our project's requirements, while Amazon RDS's managed service simplifies administration tasks, freeing up our team's resources.

In terms of schema architecture, we've opted for a star schema. This choice streamlines querying and data retrieval, vital for our analytics and reporting needs. The star schema's denormalized structure enhances performance, especially for complex datasets, aligning perfectly with our goal of facilitating efficient data analysis and empowering informed decision-making.

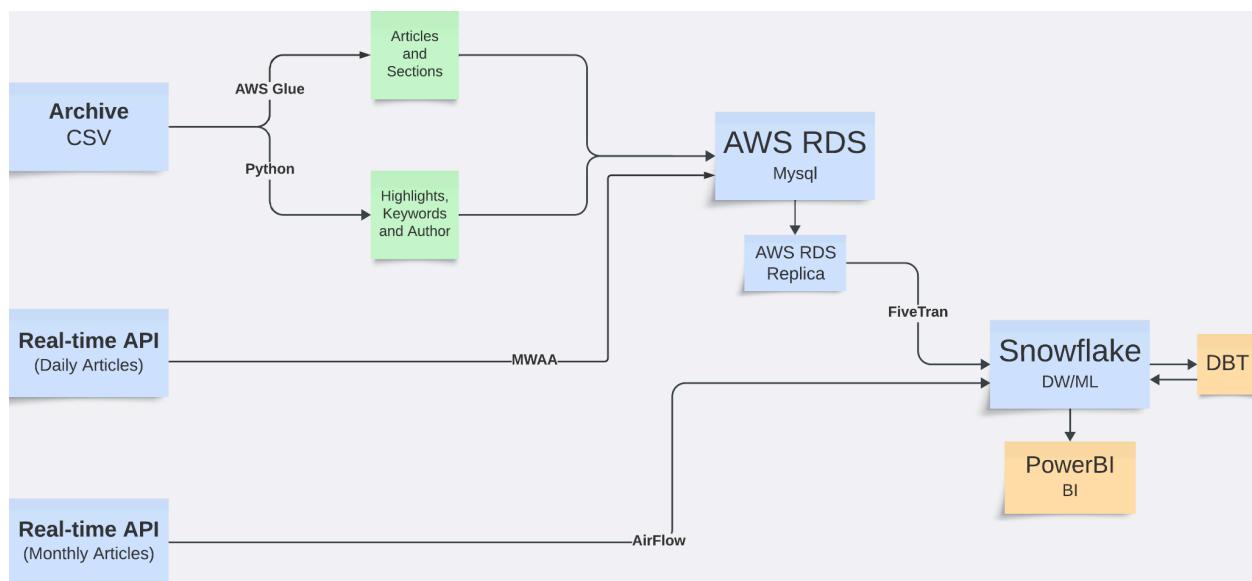
Overall, our database configuration with MySQL on Amazon RDS, coupled with the star schema architecture, reflects our commitment to optimizing data management and maximizing analytical capabilities, driving tangible business value in our cloud analytics and data warehouse implementation project.

ETLT Workflow

There are 3 workflows established in the project.

1. **Archive Data Pipeline** - The raw **Archive Data from the NY Times database** is normalized using **AWS Glue** and **Python**, loaded to **RDS** and loaded to **Snowflake**, connection established using **FiveTran**.
2. **Daily Articles Pipeline** - The **Daily Articles API** is extracted, transformed to fit in the existing schema and loaded into the same **RDS** instance, which serves as the operational database, which also eventually gets synced to **Snowflake**. This ETL from API to RDS is orchestrated by Amazon's **MWAA**. This demonstrates the incremental load.
3. **ML/BI Pipeline** - **Monthly Articles API** loads historical data from previous months. This pipeline loads the articles data month-wise to ensure no article data is missed out so that the analytics, visualization and ML analysis is updated every month and does not have any missing data. The data from this API data is directly loaded to the existing data warehouse, using **Apache Airflow**.

The data from all the three start points end up in the Data Warehouse - **Snowflake**, which is transformed using **DBT** to create Views that are helpful for Analytics and BI insights. **Visualizations** are done by connecting **PowerBI** to Snowflake and **ML models** are demonstrated using **Snowflake**.



Apache Airflow

In the context of this project, Apache Airflow serves as a robust orchestrator for managing the Extract, Transform, and Load (ETL) pipeline.

Apache Airflow employs a Directed Acyclic Graph (DAG) script architecture, which allows for the creation of complex workflows by defining tasks and their dependencies. In this project, DAGs are created to represent the ETL pipeline stages, including extracting data from real-time APIs, transforming it, and loading it into the relational database.

Data Warehouse Implementation

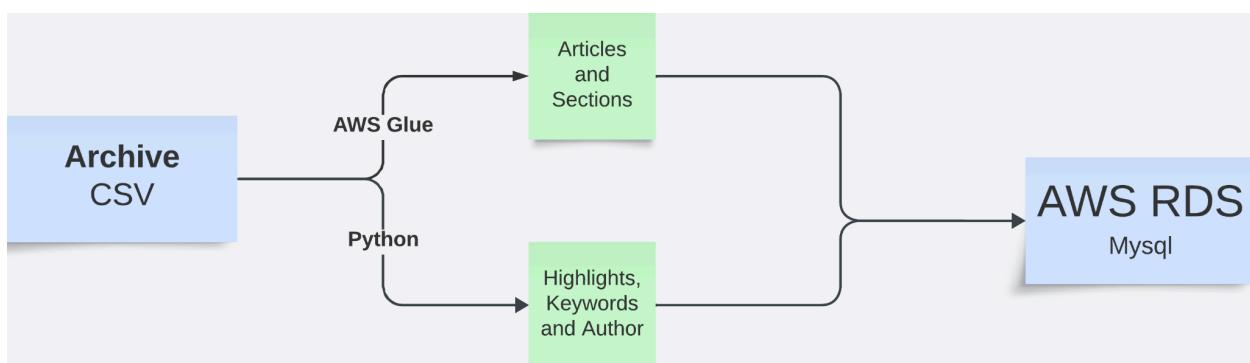
We use **Snowflake** for the Data Warehouse, where data from all the start points end up finally. All the statistical analysis and ML training happen on this data from the warehouse. To cater to the three workflows, data from three sources are configured to end up in Snowflake taking the following routes:

1. Archive Data - An initial sync of RDS data containing the Archive Data into Snowflake is done after a partner connects with FiveTran.
2. Daily Articles from Articles API - Snowflake is configured to sync with the RDS slave instance every 6 hours. The data from the replica is relayed to keep the Master DB cater to regular traffic.
3. Monthly Articles API - The data from API is transferred to Snowflake via an Airflow DAG. This is developed to ensure no article data is missed out so that the analytics, visualization and ML analysis updated every month is using the most updated information.

Workflow in Detail

1. Archive Data

a. Initial load of Archive Data to a fresh RDS instance



To address the ETL (Extract, Transform, Load) process for the archive data from the New York Times dataset, we've designed a solution that combines the power of AWS Glue for transforming the article and section data and Python for handling the transformation and loading of other related entities such as authors and keywords. Here's an overview of our approach:

1. Extraction:

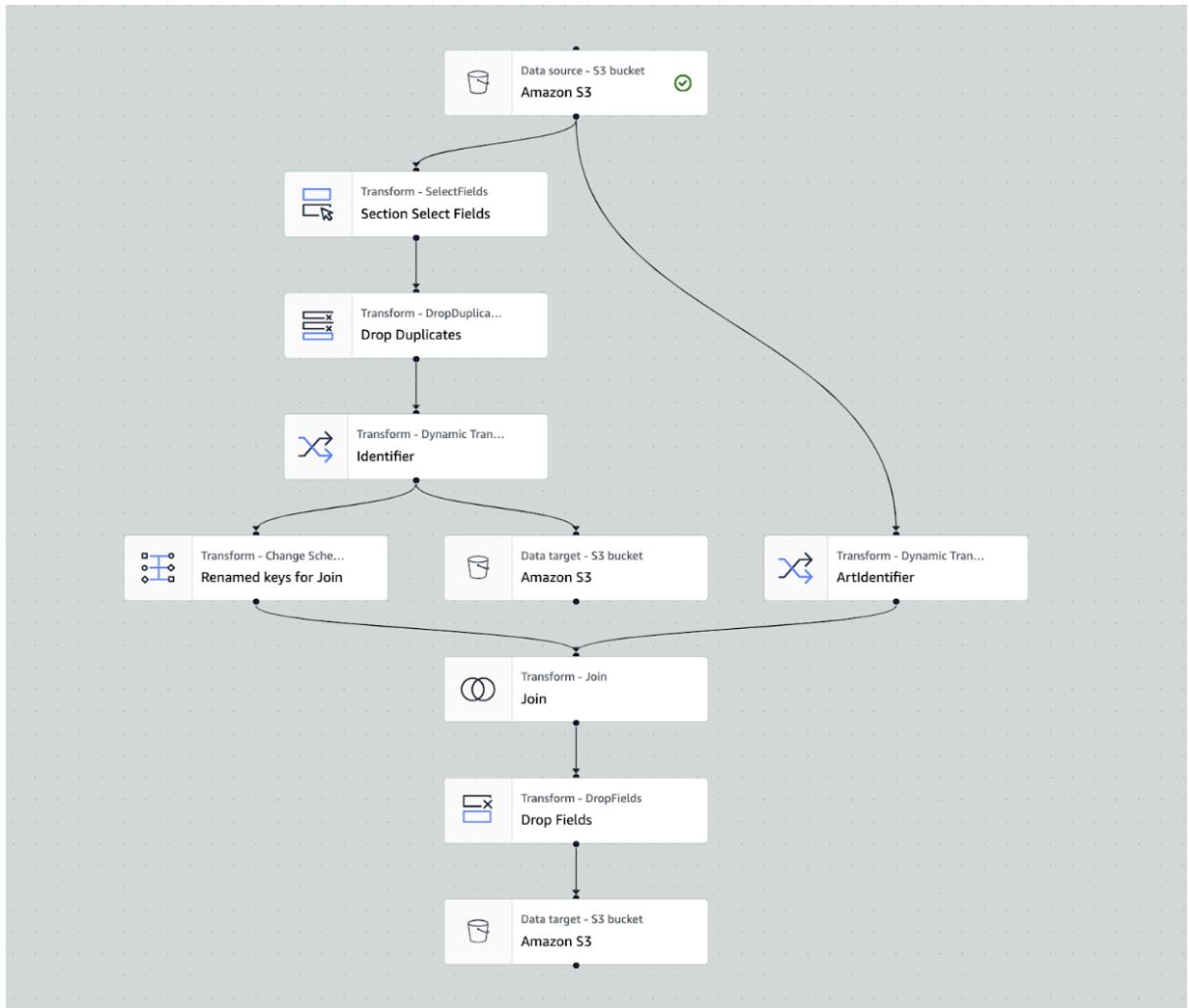
The first 100mb of the archive data from the New York Times dataset is extracted, which comprises articles, sections, headlines, authors, and keywords.

```
head -n 62000 nyt-metadata.csv > nyt.csv
```

2. Transformation:

a. For the article and section data:

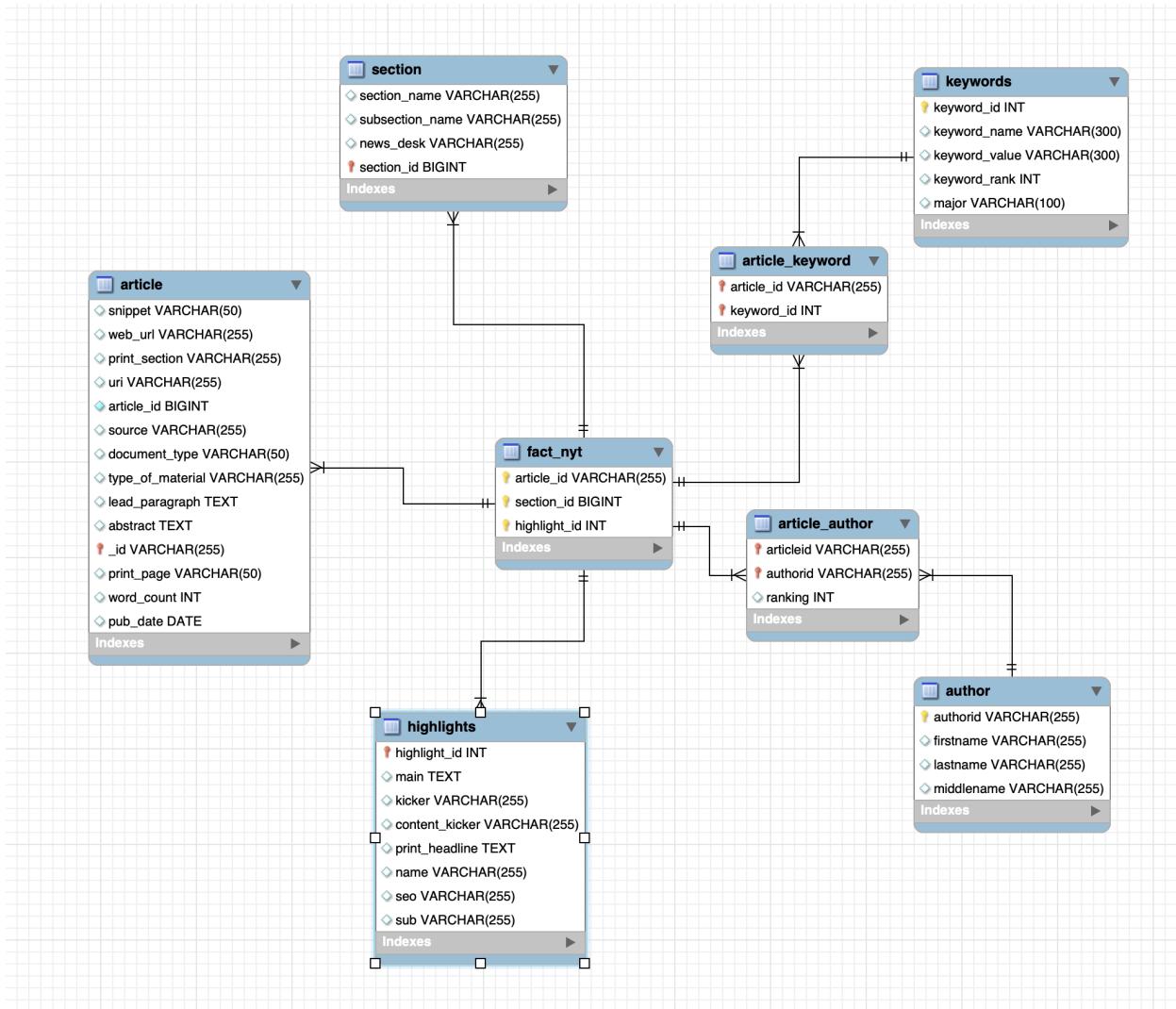
We utilize AWS Glue for transformation of article and section tables, leveraging its capabilities for easy schema mapping.



b. For authors and keywords:

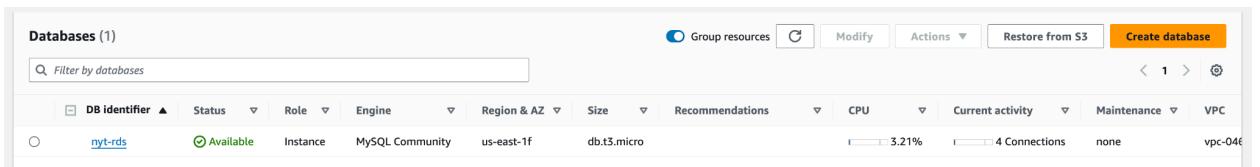
Python scripts are employed to handle the basic transformation process of headlines, keywords and authors, ensuring they conform to the specified schema. These scripts parse and extract relevant information from the raw data, performing necessary formatting as required.

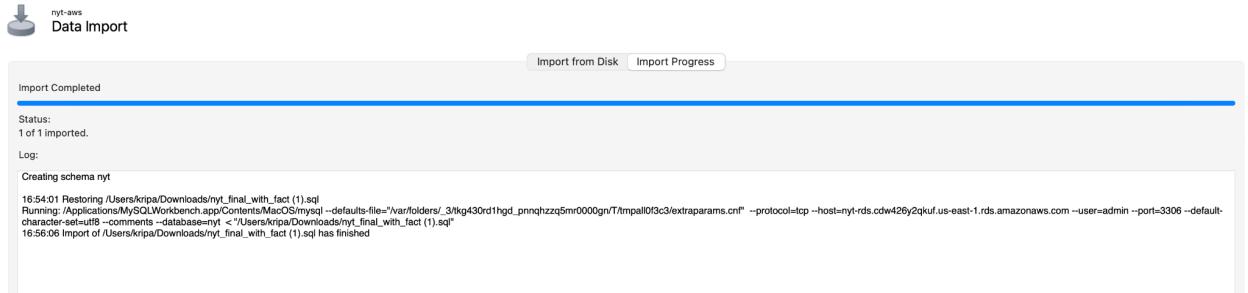
For the many-to-many relationships between articles and authors/keywords, appropriate normalization techniques are applied to ensure data integrity and efficiency. The corresponding ER diagram is shown below and the relevant code can be found on [GitHub](#).



3. Loading:

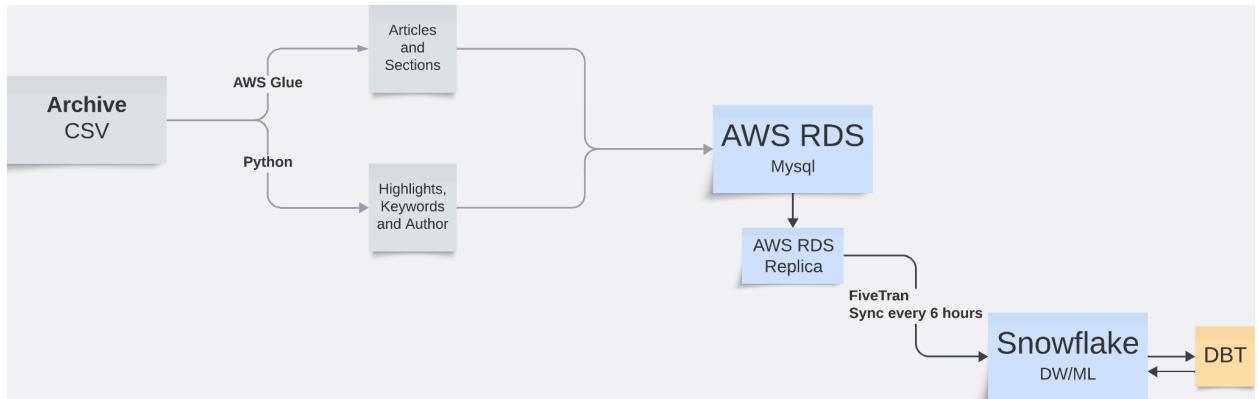
Python scripts are responsible for loading the transformed author and keyword data into the respective tables in the database, ensuring that all relationships are properly established and maintained.





b. Establish Data Warehouse and Initial Sync of Initial Archive

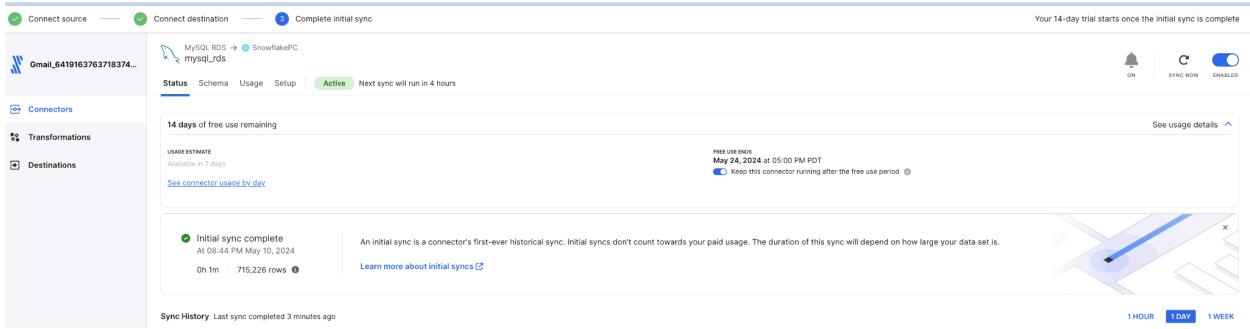
The data at Snowflake is synced from RDS through a series of steps as shown below.



1. A slave database - **Replica** is configured for the master instance at RDS. This is done to keep the operational database from any lag due to the synchronization at the backend.

Databases (2)											
DB identifier	Status	Role	Engine	Region & AZ	Size	Recommendations	CPU	Current activity	Maintenance	VPC	
nyt-rds	Available	Primary	MySQL Community	us-east-1f	db.t3.micro	3 Informational	3.21%	1 Connections	none	vpc	
nyt-read-replica	Available	Replica	MySQL Community	us-east-1c	db.t3.micro	1 Informational	3.82%	0 Connections	none	vpc	

2. A new database **NYT_DB** is created at Snowflake. A **FiveTran** partner connect is established and an initial sync of RDS is done to load the initial archive onto Snowflake's **NYT_DB**.



3. A DBT partner connect is established to create tables and views that would cater to analysis and business decisions. (An instance of ELT)

<pre> dbt-cloud-70403103924240-partner-connect-trial-repo analyses macros M models example schema.yml marts core M staging M nyt_article_pipeline Num_article_by_type.sql Top_keywords.sql article_by_date.sql nyt_article_sources.yml nyt_article_transform.sql nyt_section_keywords_transform.sql </pre>	<pre> Text_classification_input.sql article_by_date.sql author_results.sql num_article_by_type.sql nyt_article_transform.sql nyt_section_keywords_transformation.sql nyt_sources.yml region_based.sql section_authors.sql top_keywords.sql </pre>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Tables created are at NYT_RESULTS_SCHEMA:

The screenshot shows the Snowflake UI for the `NYT_DB` database. The `NYT_RESULTS_SCHEMA` is highlighted with a blue rounded rectangle. Inside the schema, there is a `Tables` folder containing nine tables: `ARTICLE_BY_DATE`, `AUTHOR_RESULTS`, `CLASSIFICATIONS_METRICS`, `CLASSIFICATIONS_RESULTS`, `NUM_ARTICLE_BY_TYPE`, `NYT_ARTICLE_TRANSFORM`, `NYT_SECTION_KEYWORDS_T...`, `SECTION_AUTHORS`, and `TEXT_CLASSIFICATION_INPUT`. Below the schema, there are also `NYT_SCHEMA` and `PUBLIC`.

```
<ul style="list-style-type: none; padding-left: 0;">- › ⏺ NYT_DB
- › ⏺ INFORMATION_SCHEMA
- › ⏺ NYT_ORIG_SCHEMA
- › ⏺ NYT_RESULTS_SCHEMA
  - › Tables
    - ⏺ ARTICLE_BY_DATE
    - ⏺ AUTHOR_RESULTS
    - ⏺ CLASSIFICATIONS_METRICS
    - ⏺ CLASSIFICATIONS_RESULTS
    - ⏺ NUM_ARTICLE_BY_TYPE
    - ⏺ NYT_ARTICLE_TRANSFORM
    - ⏺ NYT_SECTION_KEYWORDS_T...
    - ⏺ SECTION_AUTHORS
    - ⏺ TEXT_CLASSIFICATION_INPUT
    - ⏺ TOP_KEYWORDS
  - › ⏺ NYT_SCHEMA
  - › ⏺ PUBLIC

```

Views created within PC_DBT_DB:

The screenshot shows the Snowflake UI for the `PC_DBT_DB` database. Inside, there is a `DBT_DPROJECT` schema which contains a `Tables` folder and a `Views` folder. The `Views` folder contains five views: `ARTICLE_BY_DATE`, `NUM_ARTICLE_BY_TYPE`, `NYT_ARTICLE_TRANSFORM`, `NYT_SECTION_KEYWORDS_T...`, and `TOP_KEYWORDS`.

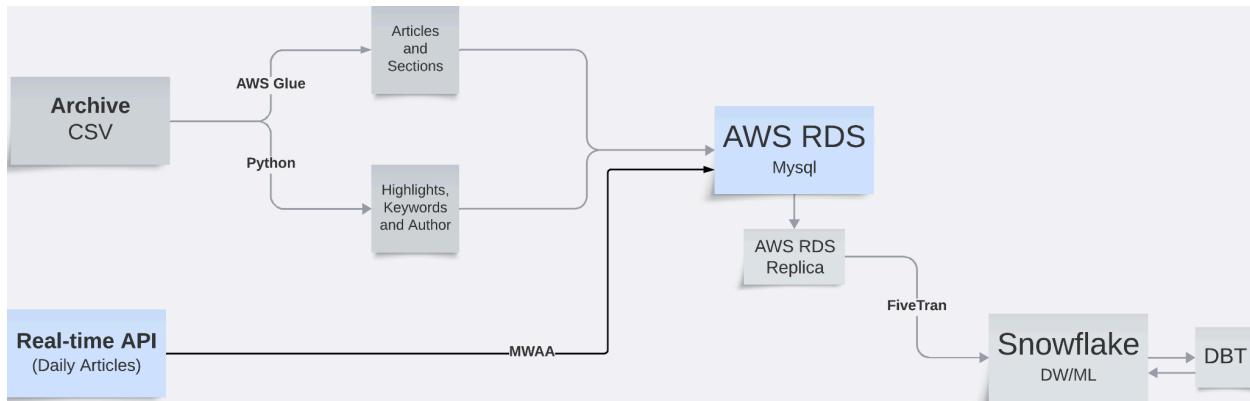
```
<ul style="list-style-type: none; padding-left: 0;">- › ⏺ PC_DBT_DB
- › ⏺ DBT_DPROJECT
  - › Tables
  - › Views
    - ⏺ ARTICLE_BY_DATE
    - ⏺ NUM_ARTICLE_BY_TYPE
    - ⏺ NYT_ARTICLE_TRANSFORM
    - ⏺ NYT_SECTION_KEYWORDS_T...
    - ⏺ TOP_KEYWORDS

```

We now have the Archive Data from the NY Times dataset, transformed and loaded into Snowflake ready for any analysis.

2. Daily Articles from Articles API

The **Daily Articles API** is extracted, transformed into a compatible schema and loaded into the same **RDS** instance. This serves as the operational database, which also eventually gets synced to **Snowflake**. This ETL from API to RDS is orchestrated by Amazon's **MWAA** and the DAG runs once everyday.



a. Daily Articles API to be used :

<https://developer.nytimes.com/docs/articlesearch-product/1/routes/articlesearch.json/get>

The API response has a list of articles on the particular day that resembles the metadata of the Archive data from the NY Times dataset. This API will be used to add the daily articles data to the RDS which already contains data from the NY Times Archive dataset.

The ETL to fetch Daily Articles from the API, make it compatible with the RDS schema and then to append to RDS MySQL DB is orchestrated by Managed Apache Airflow (MWAA).

b. Airflow Environment:

MyAirflowEnvironment is created in MWAA with its corresponding DAG code in S3://nyt4-bucket.

This screenshot shows the details of an Airflow environment named "MyAirflowEnvironment" within the AWS Managed Workflows for Apache Airflow (MWAA) service. The interface includes a navigation bar at the top with links to "Amazon MWAA", "Environments", and "MyAirflowEnvironment". On the right, there are "Edit", "Delete", and "Open Airflow UI" buttons. The main content area displays the environment's status as "Available" and provides the ARN (arn:aws:airflow:us-east-1:533267430457:environment/MyAirflowEnvironment). It also shows the Airflow UI URL (893a23b1-162f-4d3a-8a99-ac38468056b9.c49.us-east-1.airflow.amazonaws.com) and the weekly maintenance window start time (Tuesday 02:00 UTC).

The screenshot shows the 'Objects' tab in the Amazon S3 console for the 'nyt4-bucket'. The bucket contains two items:

- A folder named 'dags/'
- A text file named 'requirements.txt' with a size of 52.0 B.

A VPC with two subnets is created.

The screenshot shows the 'Networking' section of the AWS VPC console. It displays a VPC with the following details:

- Virtual private cloud (VPC): vpc-05739526cbd541f1
- Subnets:
 - subnet_01d96a8fc98f9e2a5
 - subnet_04e440f706aec920e
- Web server access: Public network
- VPC security group(s): sg-0f775078f2b86d016
- Endpoint management: Service managed endpoints

c. DAG

A DAG to fetch Daily Articles data and to push them to RDS is uploaded to nyt4-bucket/dags/load_daily_articles.py. The corresponding code is on [GitHub](#).

MySQL connection for the DAG to access RDS:

The screenshot shows the 'MySQL' connections page in the Airflow UI. It lists two connections:

Connection ID	Connection Type	Host	Port	SSL	Auth
prestashop	presto	localhost	3400	False	False
production_nyt	mysql	nyt-rds.cdw426y2qkuf.us-east-1.rds.amazonaws.com	3306	True	True

The DAG is configured to run once everyday to fetch the day's articles.

Example of a run:

The screenshot shows the 'DAG: archive_api_dag' run details in the Airflow UI. The run was triggered at 2024-05-12T00:00:00 UTC. The timeline chart shows task execution times for 'extract_dailies' and 'transform_and_load' tasks across three days.

Task Status Legend:

- defered
- failed
- queued
- removed
- restarting
- running
- scheduled
- skipped
- success
- up_for_reschedule
- up_for_retry
- upstream_failed
- no_status

Task Details:

- extract_dailies**: SUCCESS PythonOperator
- transform_and_load**: SUCCESS PythonOperator

Logs:

```
[2024-05-12, 22:10:23 UTC] {{logging_mixin.py:188}} INFO - None
[2024-05-12, 22:10:23 UTC] {{logging_mixin.py:188}} INFO - New keyword commit
[2024-05-12, 22:10:23 UTC] {{logging_mixin.py:188}} INFO - None
[2024-05-12, 22:10:23 UTC] {{logging_mixin.py:188}} INFO - New keyword commit
[2024-05-12, 22:10:23 UTC] {{logging_mixin.py:188}} INFO - None
[2024-05-12, 22:10:23 UTC] {{logging_mixin.py:188}} INFO - New keyword commit
[2024-05-12, 22:10:23 UTC] {{logging_mixin.py:188}} INFO - (27271, 'glocations', 'United States', 4, 'N', None)
[2024-05-12, 22:10:23 UTC] {{logging_mixin.py:188}} INFO - Old keyword commit
[2024-05-12, 22:10:23 UTC] {{logging_mixin.py:188}} INFO - (3917, 'subject', 'Weather', 5, 'N', None)
[2024-05-12, 22:10:23 UTC] {{logging_mixin.py:188}} INFO - Old keyword commit
[2024-05-12, 22:10:23 UTC] {{logging_mixin.py:188}} INFO - (140344, 'organizations', 'National Weather Service', 6, 'N', None)
[2024-05-12, 22:10:23 UTC] {{logging_mixin.py:188}} INFO - Old keyword commit
[2024-05-12, 22:10:23 UTC] {{logging_mixin.py:188}} INFO - New author commit
[2024-05-12, 22:10:23 UTC] {{python.py:203}} INFO - Done. Returned value was: None
[2024-05-12, 22:10:23 UTC] {{taskinstance.py:1138}} INFO - Marking task as SUCCESS. dag_id=archive_api_dag, task_id=transform_and_load, execution_date=20240512T220954, start_date=
[2024-05-12, 22:10:24 UTC] {{local_task_job_runner.py:234}} INFO - Task exited with return code 0
[2024-05-12, 22:10:24 UTC] {{taskinstance.py:3280}} INFO - 0 downstream tasks scheduled from follow-on schedule check
```

After DAG run, Daily Articles data accessed from RDS:

- d. Snowflake is configured to sync data from RDS replica every 6 hours:

Status Schema Usage **Setup** | Active Next sync will run in 3 hours**Connection Details**

Fivetran Connector ID: ovate_recant
Connected by: Gene us
bioinfo.mag@gmail.com
Connected on: May 10, 2024
Host: ryt-read-replica.cdw426y2qkuf.us-east-1.rds.amazonaws.com
Port: 3306
User: admin
Password: *****
Public Key: ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQADQDdn7xjGEKu0x8fu5yAkG96OfjjNcCbvRvIK3AB1rbAotzjflUfp2Y1ijjViUhfAJGnFniVzHjfDqn8PDV0/IRK2tpGTo4vP0CKJlj8Tn4dwMFmr1bALJvDCPAUYyy9xsj3lApGdqDuxAxVHZ33H2ZPA/gKYBAYCdTjQgBxXYWyQxF1Dkp2xpFoxTwBytqnmEqDXEQf7z/7hLqksXkwouoZWz6y+2RQdNnw5r+9eN2aWd3oSzwPHLnUK7rvvtz0vUFbN52ifmxr-CNF6Smcmjmg0nVeXhhW5mWxwXwXjP5rQLSLUs/B2+ZthlyterEILi8c5XusbApvx90RAHSVfStnML37o1wlD2jG9tcR79gkID0qQSJvwuPjupnhu/GhAfGM+idPRkQ9dhrURZtlk03VK/nQjEw5/dZQNesMmqhkZRVowE3bHn2KsVhNojkww+bkUshhdlo5ihul5TwshpYGxvArQj5vR/AoKdJJYjYpN3nmVmHP3yw+19oNNHUKjGziz5wGxk6tq2DU5qbU3EmemZpWtAA79uKI
Replica ID: 1068849452
Latest Replica Lag: 0 seconds
Update Method: BINLOG

Settings

- Sync Frequency** 6 hours
- Delay Threshold** Standard
- This connector will be considered delayed when the destination latency is greater than approx. **6.5** hours, such as when a sync takes over **30** minutes (current average duration is 41 seconds). I last successful sync.
- Re-sync All Historical Data** Click to re-sync all historical data from your source MySQL RDS connection
This can take as long to complete as the initial sync. While the re-sync is being performed, updates of new data will slow or stop, API usage will increase, and no existing data will be deleted.
- Delete Connection** Click to permanently delete the MySQL RDS connector mysql_rds.
This can not be un-done without a full re-sync.

Data at Snowflake after eventual sync:

PC_FIVETRAN_DB.MYSQL_RDS_NYT ▾ Settings ▾

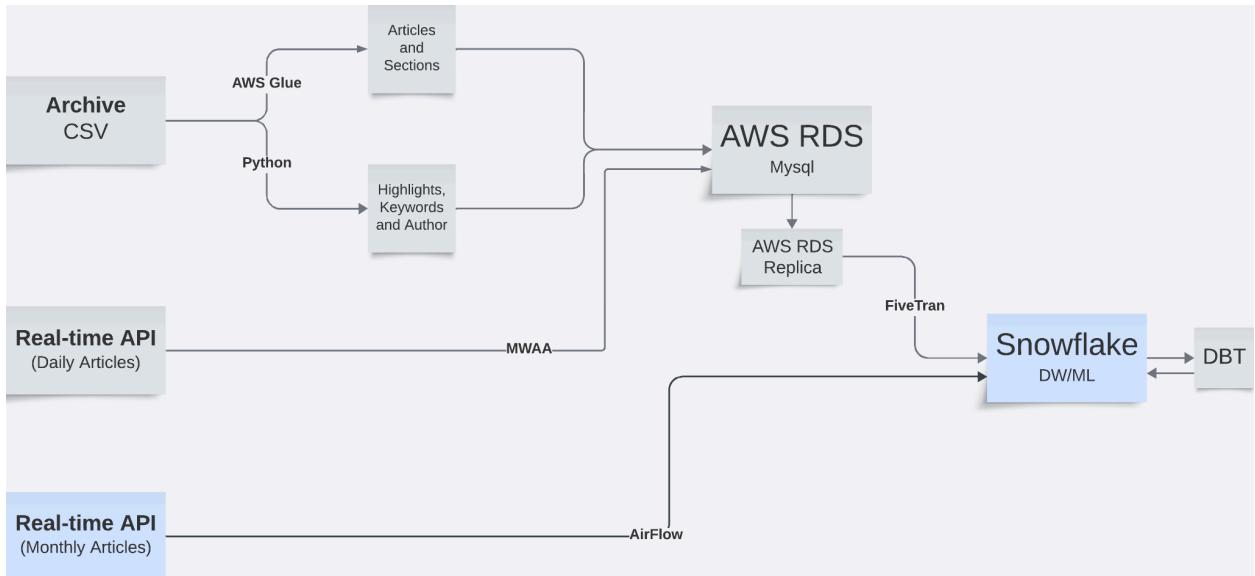
```
1 | select * from article where timestamp is not null
```

↳ Results ▾ Chart

		DOCUMENT_TYPE	_FIVETRAN_DELETED	_FIVETRAN_SYNCED	TIMESTAMP
1	marks of K-pop. No catchy tune, no snazzy outfits, no	article	TRUE	2024-05-12 08:04:16.267 +0000	2024-05-12 06:36:49.000 +0000
2	worsening humanitarian crisis in Haiti misidentified t	article	TRUE	2024-05-12 08:04:16.268 +0000	2024-05-12 06:36:50.000 +0000
3	tic team. As a child, I was not fast or coordinated or i	article	TRUE	2024-05-12 08:04:16.272 +0000	2024-05-12 06:36:46.000 +0000
4	ts are scheduled to begin in Paris, the global agency	article	TRUE	2024-05-12 08:04:16.272 +0000	2024-05-12 06:36:48.000 +0000
5	s dominated the world of B movies as the producer o	article	TRUE	2024-05-12 08:04:16.271 +0000	2024-05-12 06:36:46.000 +0000
6	of a pariah state."	article	TRUE	2024-05-12 08:04:16.268 +0000	2024-05-12 06:36:51.000 +0000
7	Theme	article	TRUE	2024-05-12 08:04:16.190 +0000	2024-05-12 01:42:50.000 +0000
8	ten about his faith, acknowledged on Saturday that I	article	TRUE	2024-05-12 08:04:16.194 +0000	2024-05-12 01:19:34.000 +0000
9	irovision Song Contest final in Malmo, Sweden, was	article	TRUE	2024-05-12 08:04:16.191 +0000	2024-05-12 01:24:56.000 +0000
10	: with student protest, the graduation ceremony at t	article	TRUE	2024-05-12 08:04:16.192 +0000	2024-05-12 01:40:33.000 +0000
11	s southern and northern Gaza are being forced to flee	article	TRUE	2024-05-12 08:04:16.193 +0000	2024-05-12 01:56:09.000 +0000
12	keley, hundreds of soon-to-be graduates rose fro	article	TRUE	2024-05-12 08:04:16.193 +0000	2024-05-12 01:27:07.000 +0000
13	elevision station KTLA 5 in Los Angeles whose morn	article	TRUE	2024-05-12 08:04:16.192 +0000	2024-05-12 01:49:02.000 +0000

We now have the Daily Articles data from API persisted in RDS via MWAA DAG, which gets eventually synced to Snowflake.

3. Monthly articles data from Archive API



The API returns an array of NYT articles for a given month and year. Its response fields are the same as the Article Search API which is loaded on a daily basis. Updating snowflake warehouse with data on a monthly basis on top of the data from Article Search API will ensure no article data is missed out due errors or down times. After updating the data received from API the data transformations are applied to generate result tables necessary for analytics, visualizations and ML analysis. This is achieved by implementing DAG pipelines using Apache Airflow.

The screenshot shows the Airflow web interface with the following details:

- DAGs** tab is selected.
- Filter: Active (1), Paused (0).
- Running (0), Failed (0).
- Search: Filter DAGs by tag, Search DAGs.
- Actions: Auto-refresh, Refresh.
- DAGs listed:
 - ml_classification_pipeline**: Owner airflow, Runs 1, Schedule 0 12 1 * * 0, Last Run 2024-05-12, 16:45:13, Next Run 2024-06-01, 12:00:00, Recent Tasks 1.
 - real_time_api_pipeline**: Owner airflow, Runs 1, Schedule @monthly, Last Run 2024-05-12, 06:56:19, Next Run 2024-06-01, 00:00:00, Recent Tasks 2.
 - transformation_pipeline**: Owner airflow, Runs 1, Schedule 0 8 1 * * 0, Last Run 2024-05-12, 07:01:28, Next Run 2024-06-01, 06:00:00, Recent Tasks 6.
- Pagination: Showing 1-3 of 3 DAGs.

Real Time API Processing pipeline:

A DAG pipeline with python operators fetches the previous month articles data from NYT Archive API, processes and loads it to Snowflake warehouse if there are any new articles found. During the data processing step the data is extracted and formatted into tables to match the schema of Snowflake Warehouse. This pipeline is scheduled to run every 1st day of the month at 12 am. The pipeline fetches articles from the current year and previous month.

DAGs

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
ml_classification_pipeline	airflow	1	0 0 * * * Schedule: At 00:00, on day 1 of the month	2024-05-12, 16:45:13	2024-05-01, 12:00:00	1	▶ ⟳	...
real_time_api_pipeline	airflow	2	0 1 * * * monthly	2024-05-12, 06:56:19	2024-06-01, 00:00:00	1 2	▶ ⟳	...
transformation_pipeline	airflow	2	0 1 * * * daily	2024-05-12, 07:01:28	2024-06-01, 06:00:00	1 2	▶ ⟳	...

Showing 1-3 of 3 DAGs

DAG: real_time_api_pipeline Fetch data from New York Times API and load into Snowflake

Schedule: @monthly | Next Run ID: 2024-06-01, 00:00:00 UTC | ▶ ⟳

05 / 12 / 2024 06 : 56 : 19 AM All Run Types All Run States Clear Filters Auto-refresh 25

Duration: 00:09:41

May 12, 2024

May 13, 2024

preprocess_data | load_data

Details Graph Gantt Code Audit Log

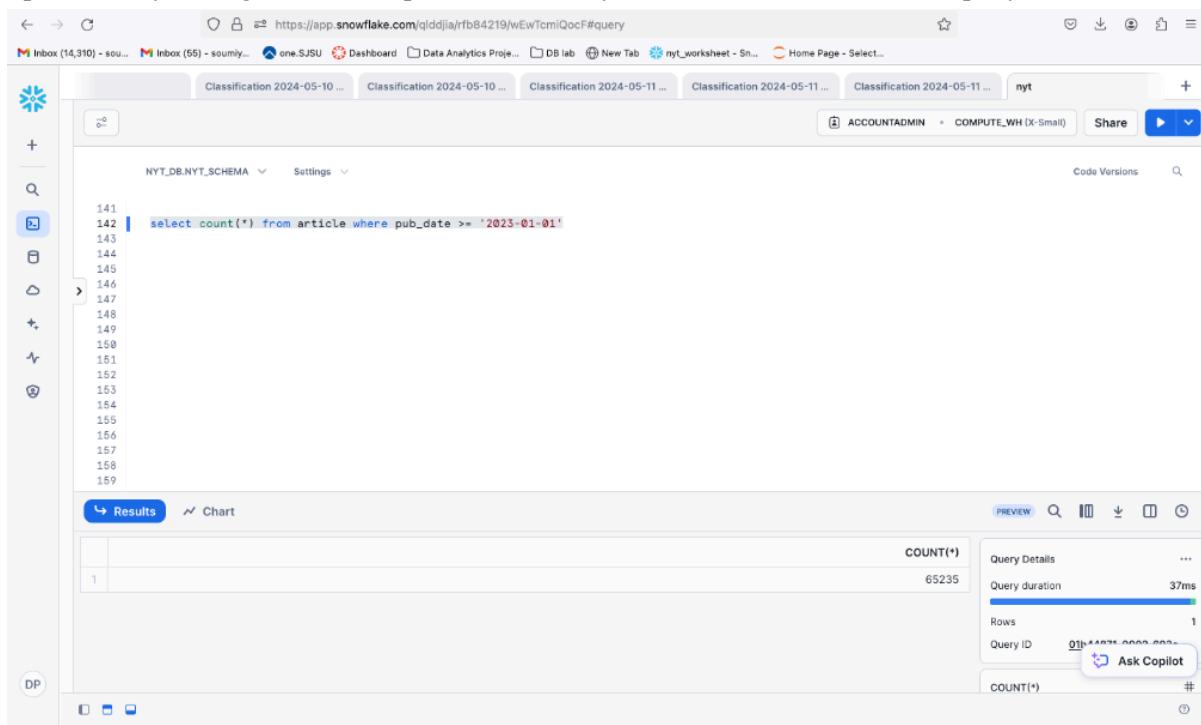
Layout: Left -> Right

preprocess_data: success PythonOperator

load_data: success PythonOperator

Also, this pipeline was manually triggered to load some of the historic data missing in the database. The number of articles inserted into the Snowflake database NYT_DB.NYT_SCHEMA from January 2023 to

April 2024 by calling the API to update the monthly articles data is shown in the query result below:



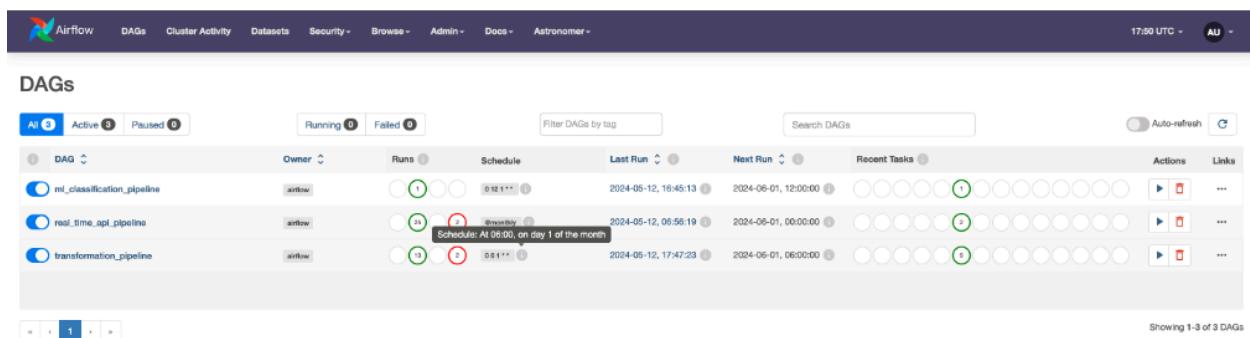
The screenshot shows a Snowflake worksheet interface. The URL is https://app.snowflake.com/qjddjia/rfb84219/wEwTcmiQocF#query. The query is:

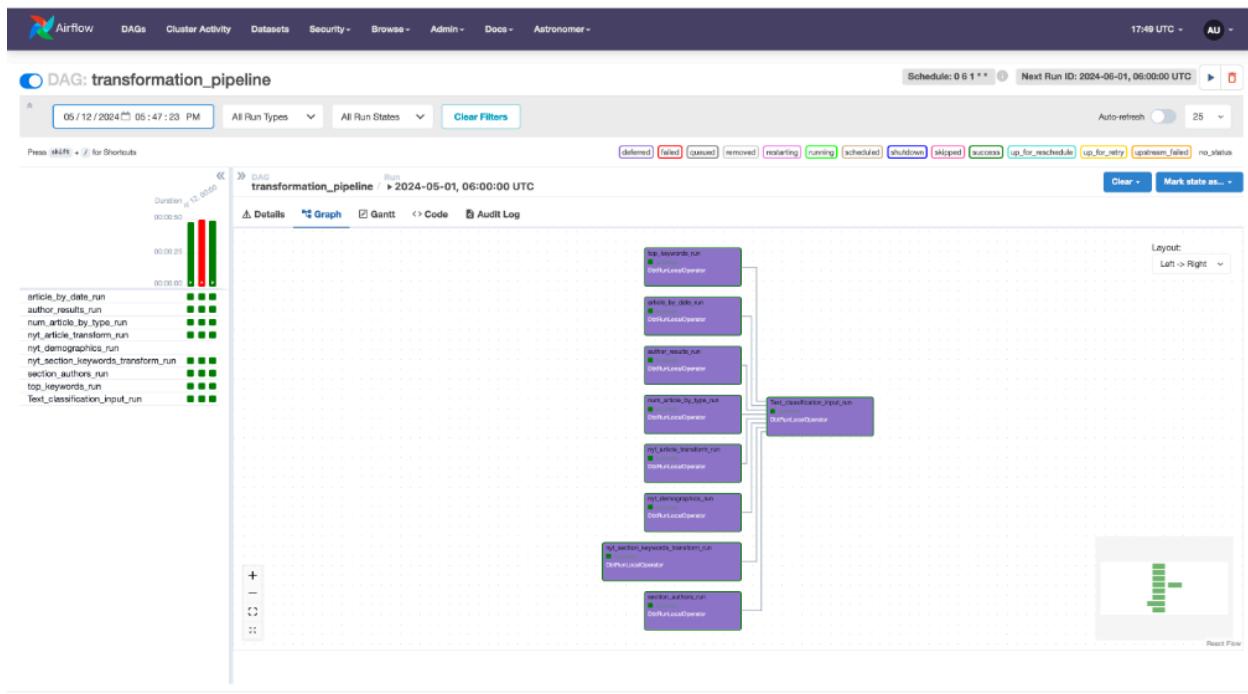
```
NYT_DB.NYT_SCHEMA
141
142 select count(*) from article where pub_date >= '2023-01-01'
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
```

The results pane shows a single row with COUNT(*) = 65235. The Query Details panel indicates a duration of 37ms, 1 row, and a Query ID of 01b44675-0000-4023-8000-000000000000. An "Ask Copilot" button is visible in this panel.

Transformation Pipeline:

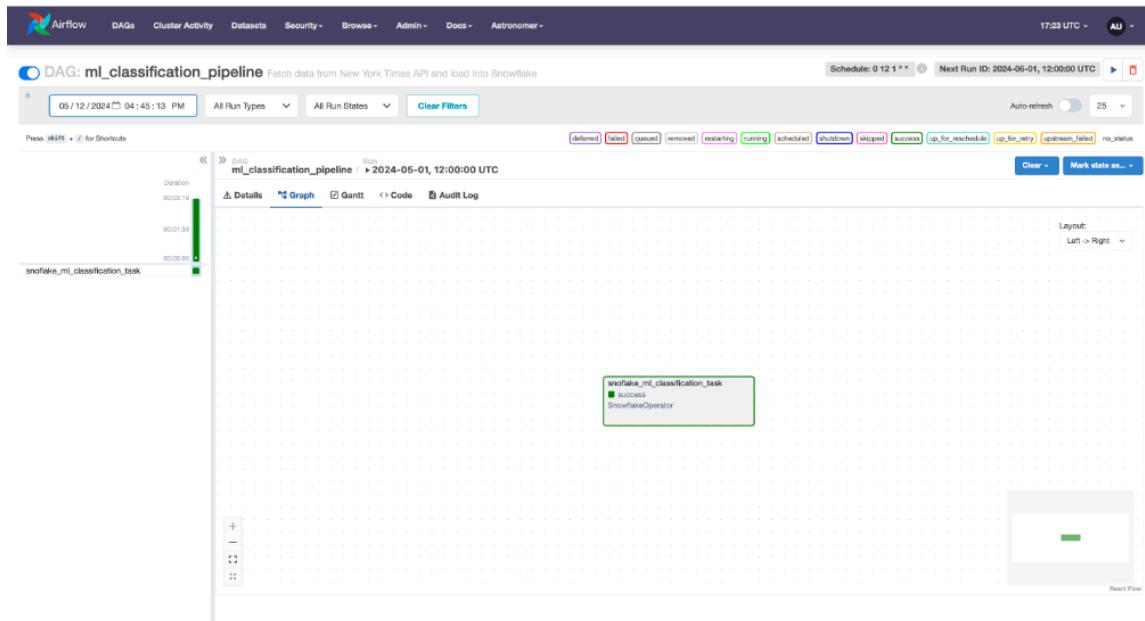
A DBTDAG pipeline performs dbt transformations and generates summarized result tables for analytics, visualizations and ML analysis. The result tables are stored in the Snowflake database NYT_DB.NYT_RESULTS_SCHEMA. This pipeline is scheduled to run every 1st day of the month at 6 am. This is scheduled after the Real Time API pipeline, so that the transformations are applied for updated databases and the results are kept ready for Visualizations and ML.





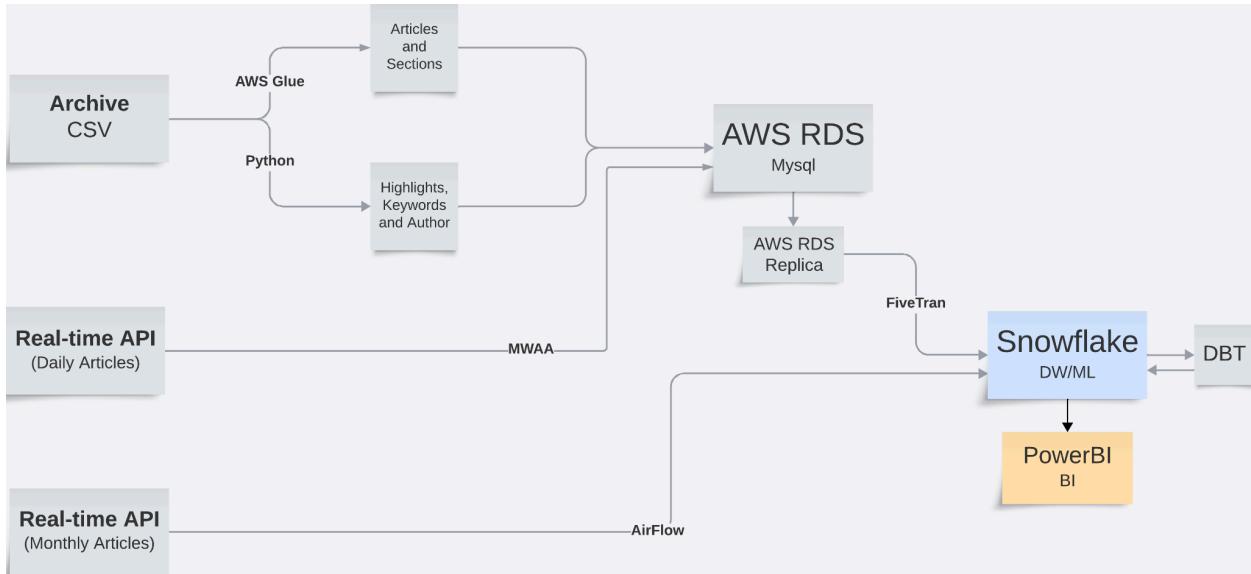
Machine Learning Text Classification Pipeline:

A DAG pipeline with snowflake operator which performs training and evaluation of data using Snowflake ML Classifications Algorithm to predict section name of articles. This pipeline uses the transformed results developed in the Transformation pipeline for ML analysis, stores the classification results and classification metrics in the Snowflake database NYT_DB.NYT_RESULTS_SCHEMA. This pipeline is scheduled to run every 1st day of the month at 12 pm.



The code for the scheduled pipelines: [Github](#)

Business Intelligence



Connect Power BI and Snowflake

Steps to connect using basic connection:

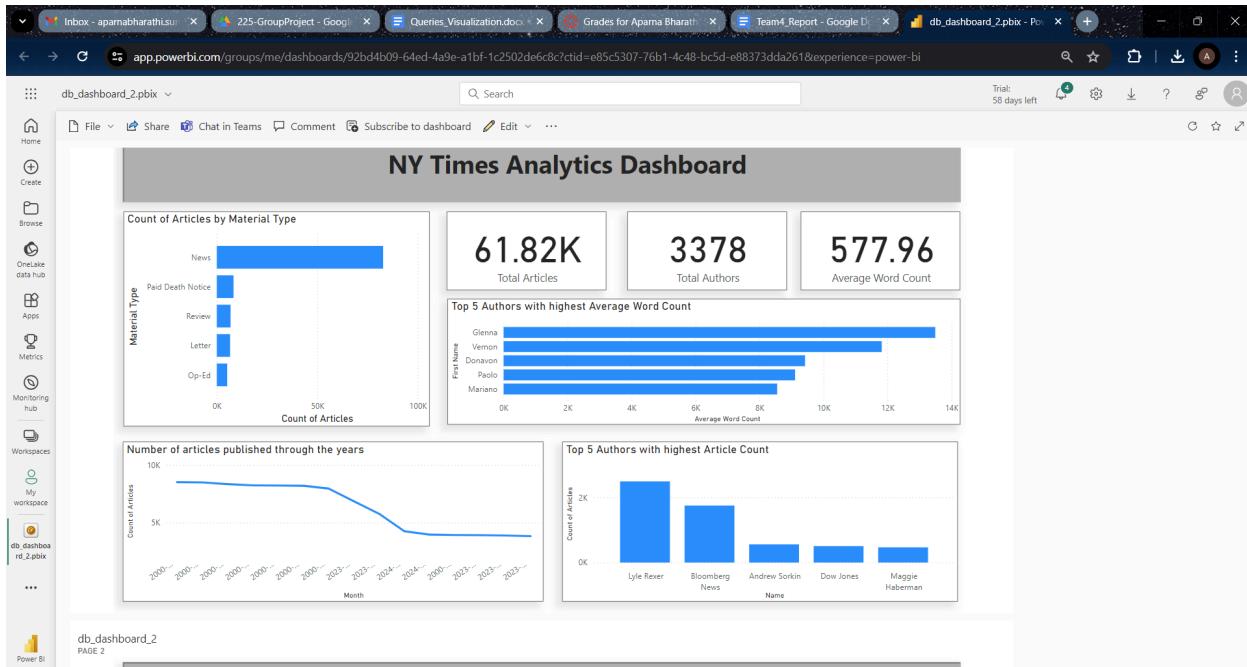
1. Open Power BI Desktop and click "Get Data" from the Home ribbon.
2. In the "Database" category, select "Snowflake" and then "Connect".
3. Enter the Snowflake server address and warehouse name.
4. Provide your Snowflake username and password and click "Sign in".
5. Choose the tables or views you want to import or use DirectQuery to connect to the live data.

Connection Mode: When connecting, you can choose between Import and DirectQuery. Import brings the data into Power BI for offline analysis, while DirectQuery connects live to Snowflake for real-time data.

Please find the screenshots for all the above steps [here](#).

Power BI Dashboard

Dashboard View:



We created 2 dashboards, Dashboard 1 has the summary of all the reports and dashboard 2 has a detailed analysis. Please find the links below:

[Dashboard 1](#) & [Dashboard 2](#)

Statistical Analysis

We utilize SQL queries executed through DBT on the data present in Snowflake to conduct a comprehensive statistical analysis of article data. We aim to provide actionable insights into readership preferences, content trends, and keyword usage:

- Analyze which region has the most number of articles(America vs other other Continents).
- Analyze the number of articles per month and year(Compare 2000 and 2023).
- Evaluate the top 5 keywords used in each group(Eg:Organizations,Subject,persons..).
- Analyze the number of articles by the type of content(Eg:News, Letter,Paid Death Notice..)
- Analyze keywords associated with popular articles to understand reader interests and preferences.
- Count the number of words in each section(Eg.Sports,Arts).
- Analyze the Distribution of keywords across different sections of articles.
- Evaluate the Correct and incorrect PREDICTIONS by the group(Eg:Food,Arts..)
- Determine which sections are attracting the most attention and which may need improvement or more focus.

Please find the queries used for statistical analysis [here](#).

Please find the Corresponding DBT Models [here](#)

Please find the visualization screenshots [here](#).

Machine Learning and Recommendations:

We've leveraged Snowflake's ML Text Classifier, employing machine learning and NLP for efficient text categorization. It processes extensive text data, extracting essential features to ensure precise classification. Integrated into Snowflake's scalable platform, it streamlines the management of textual data seamlessly.

The process involves feeding the abstract (a summary), the lead paragraph (the opening paragraph), and information about the news desk (the department or team within a publication responsible for certain topics) of an article into a model. The model then uses this information to automatically categorize the article into relevant sections, providing recommendations on which section it belongs to within a publication. Going forward, the articles needn't have any section data and the model would be able to categorize appropriately. This helps streamline the organization and distribution of articles by ensuring they are placed in appropriate sections for readers.

Table showing the model results for section name prediction:

The screenshot shows the Snowflake Data Analytics workspace interface. The top navigation bar includes tabs for 'Classification 2024-05-10 ...' through 'Classification 2024-05-11 ...'. The main area displays a query result table titled 'NYT_DB.NYT_RESULTS_SCHEMA'. The table has columns: ABSTRACT, SECTION_NAME, LEAD_PARAGRAPH, NEWS_DESK, PREDICTIONS, PREDICTED_SECTION_NAME, and PROBABILITY. The results show various news items categorized into Health, Science, and Arts. A sidebar on the left provides navigation and search functions. On the right, there is a 'Query Details' panel showing metrics like duration (82ms), rows (179.4K), and query ID (01b4487d-0002-6939-...). An 'Ask Copilot' button is also visible.

ABSTRACT	SECTION_NAME	LEAD_PARAGRAPH	NEWS_DESK	PREDICTIONS	PREDICTED_SECTION_NAME	PROBABILITY
1 Helping someone in	Health	Before 1958, there was	Science	{ "class": "Health", "lo	Health	0.997
2 Learning delays an	Health	Children experienced i	Science	{ "class": "Health", "lo	Health	0.997
3 Many Black women	Health	Shakima Tozay was 37	Science	{ "class": "Science", "l	Science	0.569
4 The justices are po	Health	Less than a year after	Science	{ "class": "Science", "l	Science	0.569
5 Officials have share	Health	In the month since fed	Science	{ "class": "Science", "l	Science	0.569
6 Philadelphia is agai	Health	Quetcy M. Lozada, a fi	Science	{ "class": "Science", "l	Science	0.569
7 Results of a new st	Health	MDMA-assisted therap	Science	{ "class": "Science", "l	Science	0.569
8 States across the c	Health	States across the cour	Health	{ "class": "Health", "lo	Health	0.996
9 The F.D.A. is invest	Health	With dozens of childre	Science	{ "class": "Science", "l	Science	0.569
10 The federal agency	Health	Federal regulators said	Science	{ "class": "Health", "lo	Health	0.997
11 The products, offer	Health	The Food and Drug Ad	Science	{ "class": "Science", "l	Science	0.569
12 "It is really hard," a	Health	Claire M. Fagin, a leadi	Obits	{ "class": "Arts", "log	Arts	0.302
13 Researchers at Sta	Health	Within every cancer ar	Science	{ "class": "Science", "l	Science	0.569

Accuracy is a metric used to evaluate the performance of a model. It represents the proportion of correct predictions made by the model out of the total predictions. These counts help assess the model's effectiveness in making accurate predictions.

Table showing the classification metrics for section name prediction:

The screenshot shows a Snowflake query results page. The URL is https://app.snowflake.com/qjddjia/rfb84219/wEwTcmIQocF#query. The query is:

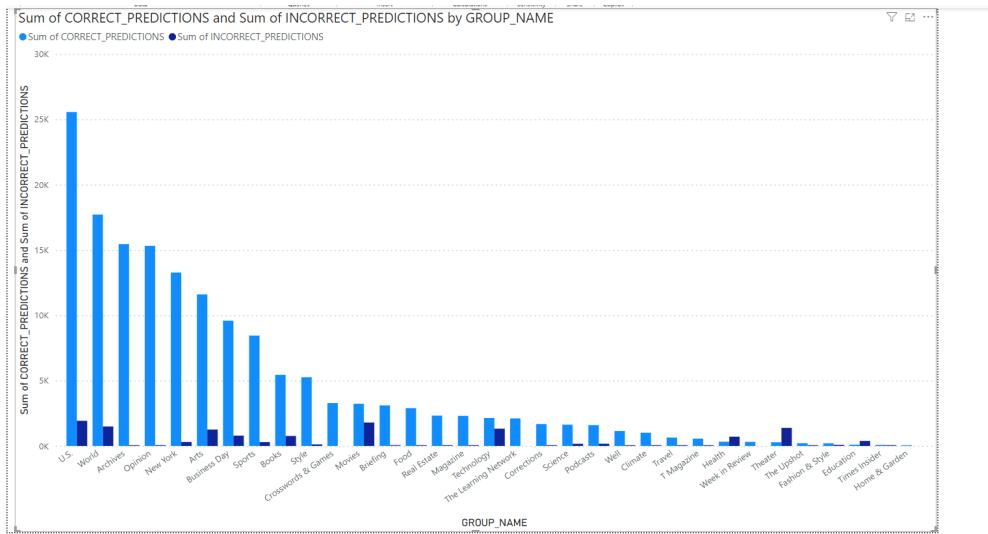
```
145 | select * from classifications_metrics order by correct_predictions desc;
```

The results table has columns: GROUP_NAME, ACTUAL_COUNT, CORRECT_PREDICTIONS, INCORRECT_PREDICTIONS, and ACCURACY. The data is as follows:

GROUP_NAME	ACTUAL_COUNT	CORRECT_PREDICTIONS	INCORRECT_PREDICTIONS	ACCURACY
1 U.S.	28746	26702	2044	92.89
2 World	20227	18596	1631	91.94
3 Opinion	15962	15922	40	99.75
4 Archives	15500	15448	52	99.66
5 New York	13901	13563	338	97.57
6 Arts	13503	12161	1342	90.06
7 Business Day	10905	10054	851	92.20
8 Sports	8781	8453	328	96.26
9 Books	6458	5677	781	87.91
10 Style	5661	5537	124	97.81
11 Crosswords & Games	3536	3536	0	100.00
12 Movies	5214	3325	1889	63.77
13 Briefing	3276	3262	14	99.57

On the right side, there are sections for Query Details (duration 305ms, rows 33, query ID 01b4487b-0002-693a-...), a histogram for ACTUAL_COUNT, and a histogram for CORRECT_PREDICTIONS with an "Ask Copilot" button.

Visual showing the Number of Correct Predictions vs Incorrect Predictions:



References

Github

<https://github.com/soumiyarao/data225-project/>

Query:

<https://acrobat.adobe.com/link/review?uri=urn:aaid:scds:US:fdea6c50-316e-310e-a0ce-e665537fcf6b>

Statistical Analysis/BI

Connect PowerBI and Snowflake :

<https://acrobat.adobe.com/link/review?uri=urn:aaid:scds:US:79604511-edb5-38db-a473-8abb20645ada>

Visualizations :

<https://acrobat.adobe.com/link/review?uri=urn:aaid:scds:US:5135719c-4962-332f-92bd-d177f2aa6497>

Power BI Dashboard:

Dashboard 1:

https://app.powerbi.com/groups/me/dashboards/92bd4b09-64ed-4a9e-a1bf-1c2502de6c8c?ctid=e85c5307-76b1-4c48-bc5d-e88373dda261&pbi_source=linkShare

Dashboard 2:

https://app.powerbi.com/groups/me/dashboards/090b926c-612d-410a-9fe6-3adf52ab39ec?ctid=e85c5307-76b1-4c48-bc5d-e88373dda261&pbi_source=linkShare

Others

Daily Articles API : <https://developer.nytimes.com/docs/articlesearch-product/1/types/Article>

API for Monthly Articles : <https://developer.nytimes.com/docs/archive-product/1/types/Article>