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.

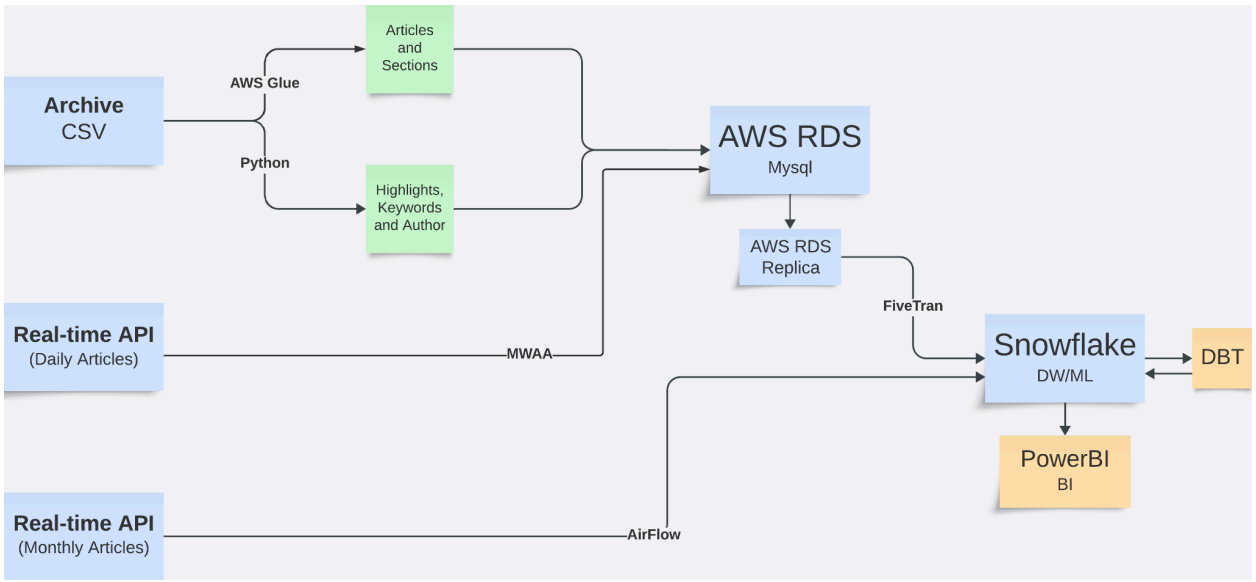
Overall, our database configuration with MySQL on Amazon RDS, coupled with the star schema architecture, reflects our commitment to optimizing data management and maximizing analytical capabilities, driving tangible business value in our cloud analytics and data warehouse implementation project.

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

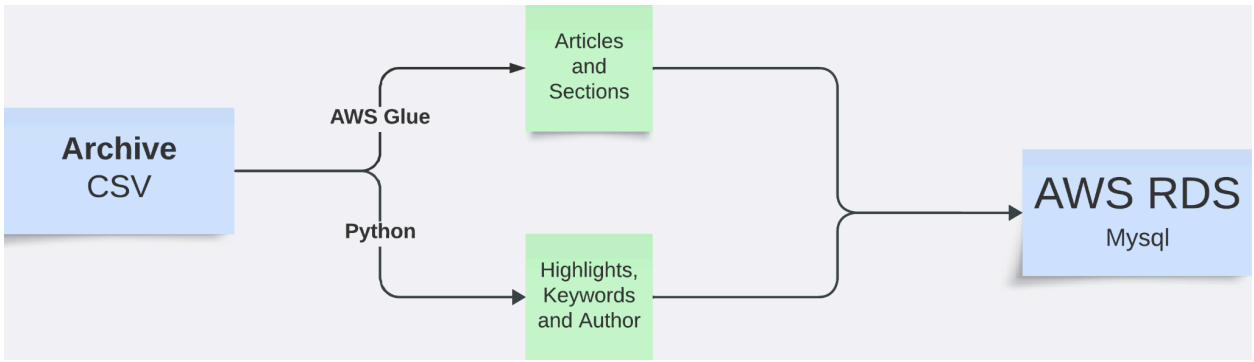
We use **Snowflake** for the Data Warehouse, where data from all the start points end up finally. All the statistical analysis and ML training happen on this data from the warehouse. To cater to the three workflows, data from three sources are configured to end up in Snowflake taking the following routes:

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

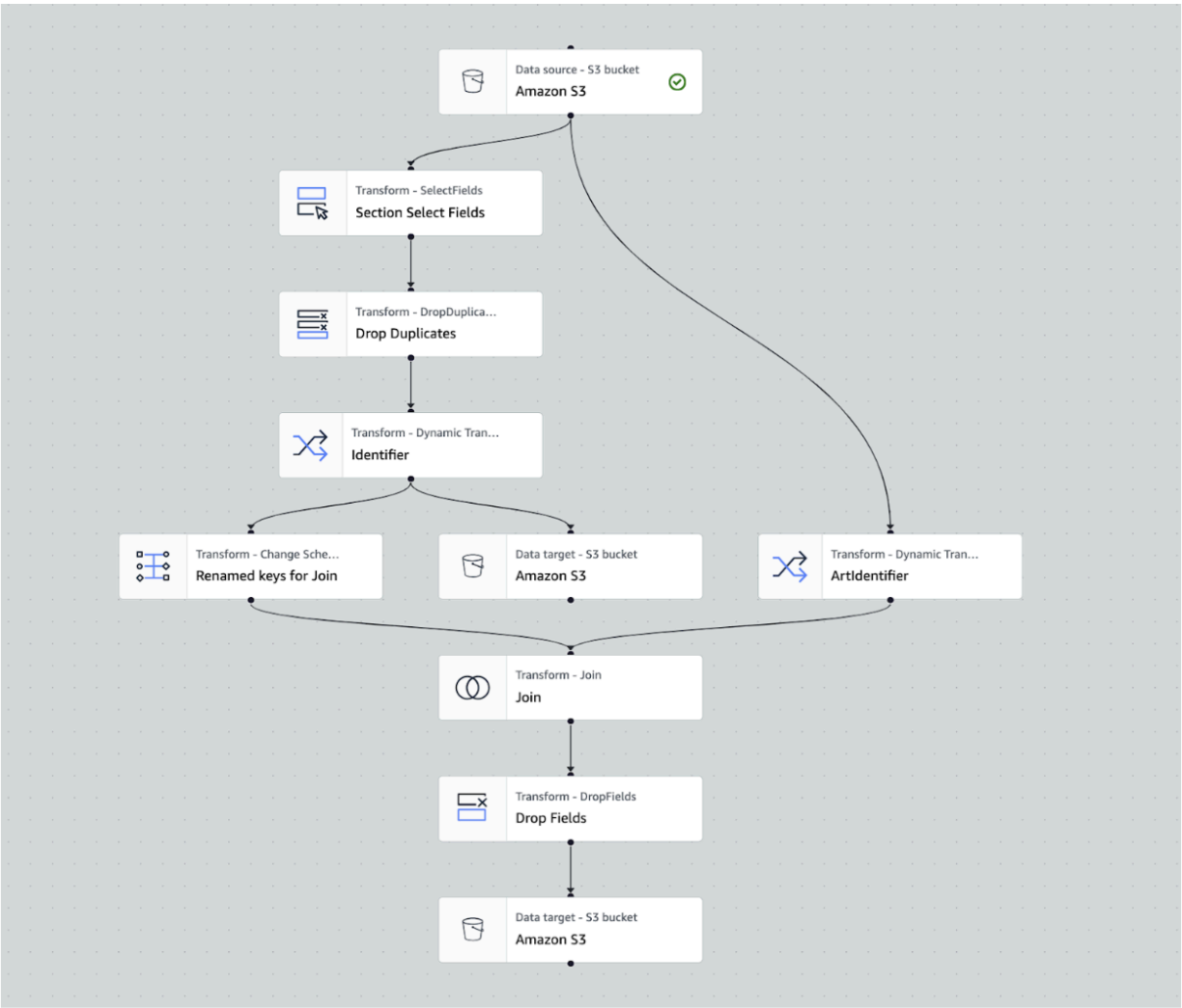


To address the ETL (Extract, Transform, Load) process for the archive data from the New York Times dataset, we've designed a solution that combines the power of AWS Glue for transforming the article and section data and Python for handling the transformation and loading of other related entities such 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`

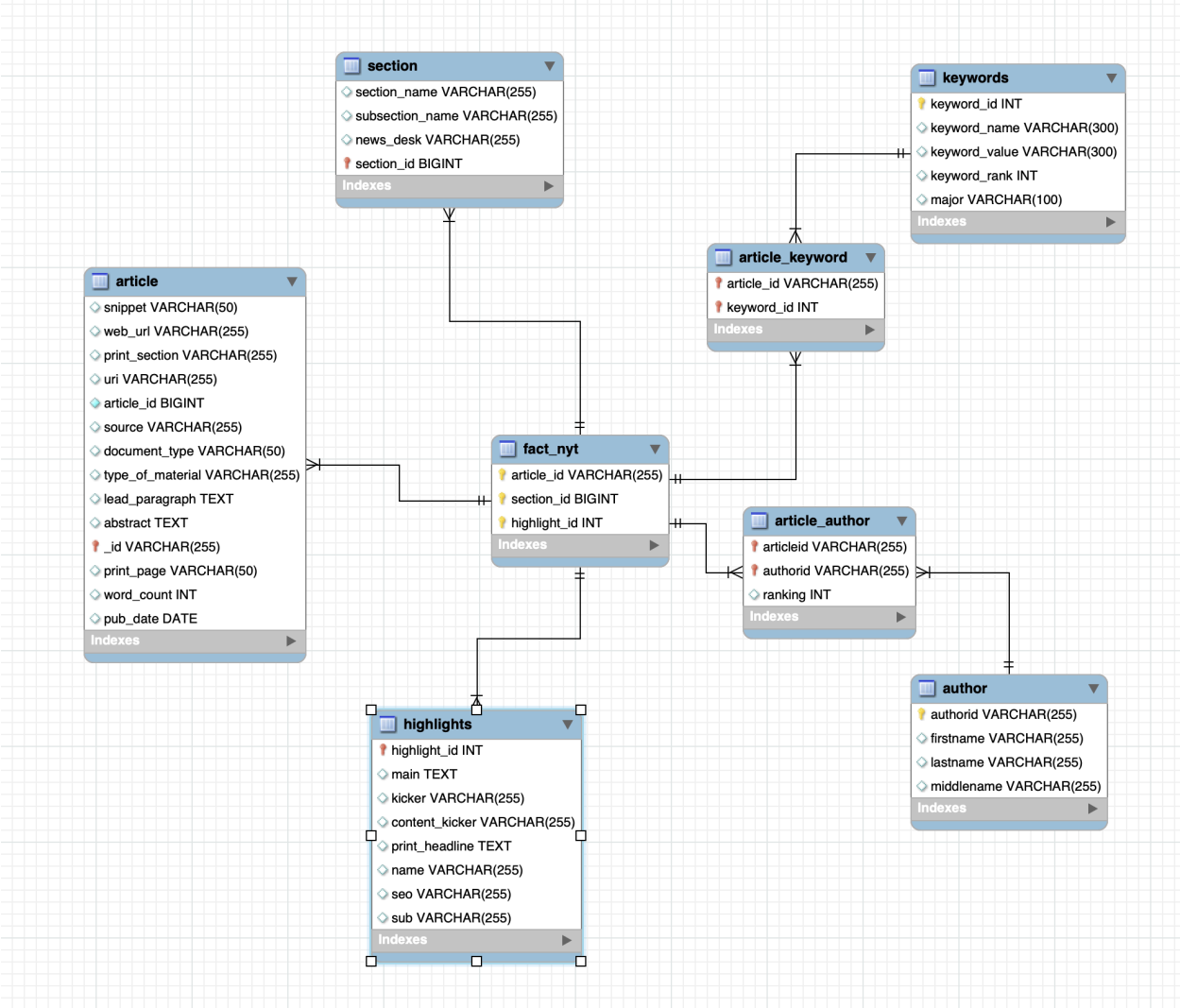
2. **Transformation:**

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.

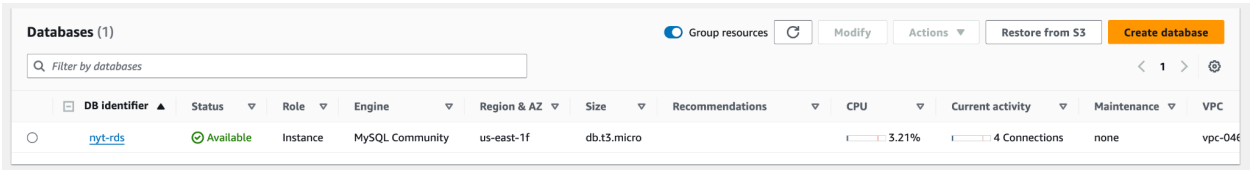


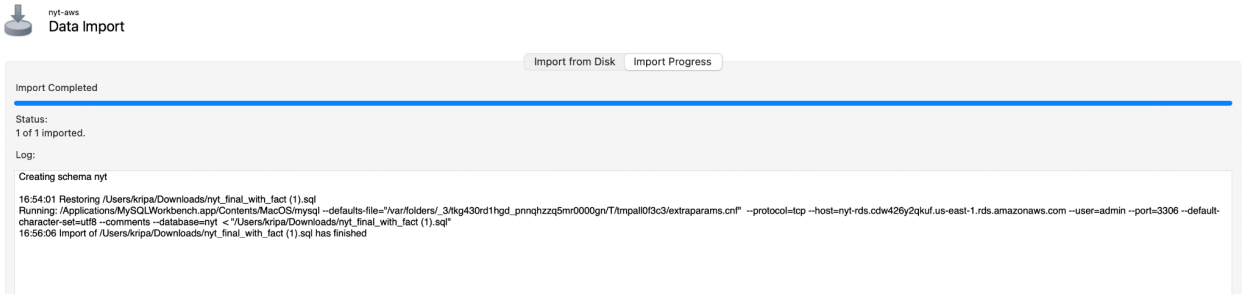
b. **For authors and keywords:**
   Python scripts are employed to handle the basic transformation process of headlines, keywords and authors, ensuring they conform to the specified schema. These scripts parse and extract relevant information from the raw data, performing necessary formatting as required.

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.



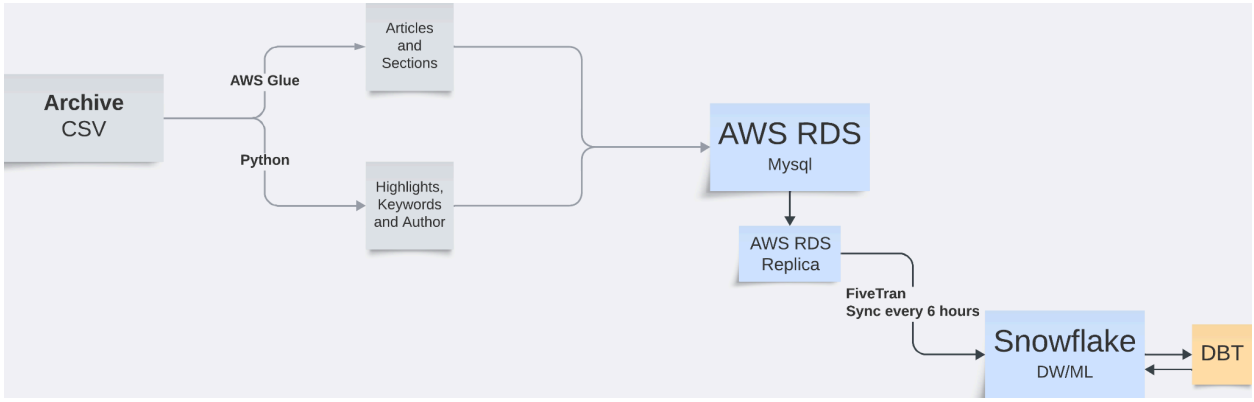
3. **Loading:**
   Python scripts are responsible for loading the transformed author and keyword data into the respective tables in the database, ensuring that all relationships are properly established and maintained.



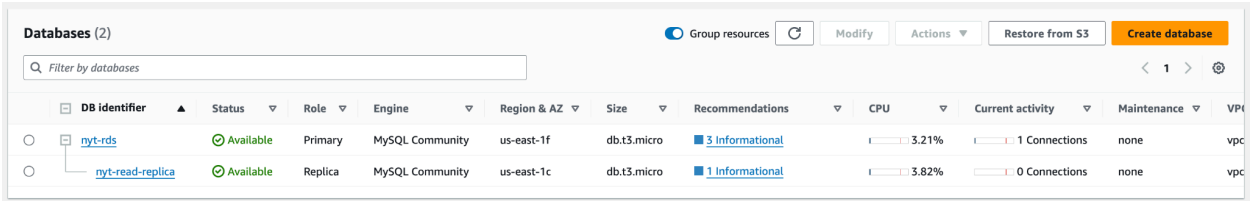


b. Establish Data Warehouse and Initial Sync of Initial Archive

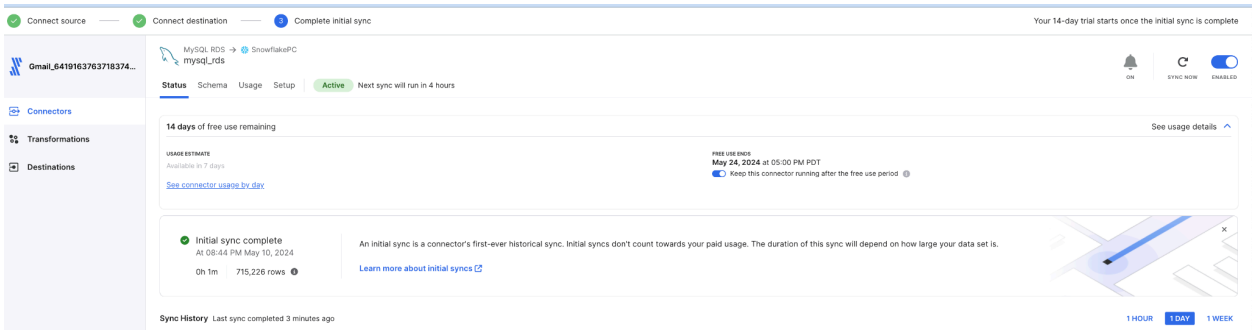
The data at Snowflake is synced from RDS through a series of steps as shown below.



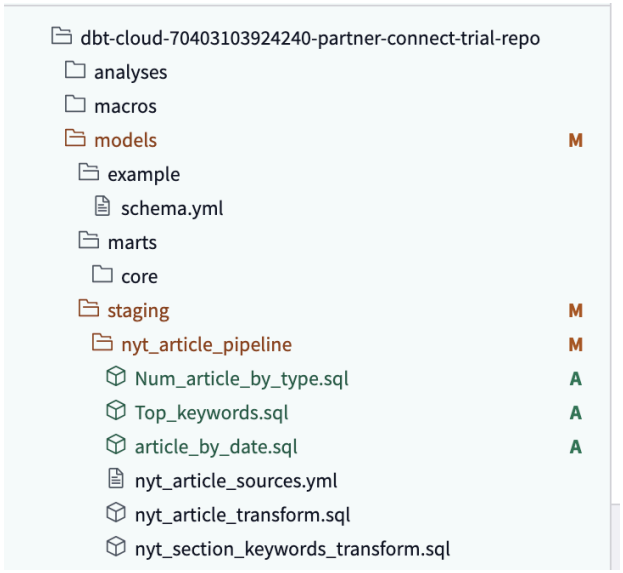
1. *A slave database - **Replica** is configured for the master instance at RDS. This is done to keep the operational database from any lag due to the synchronization at the backend.*

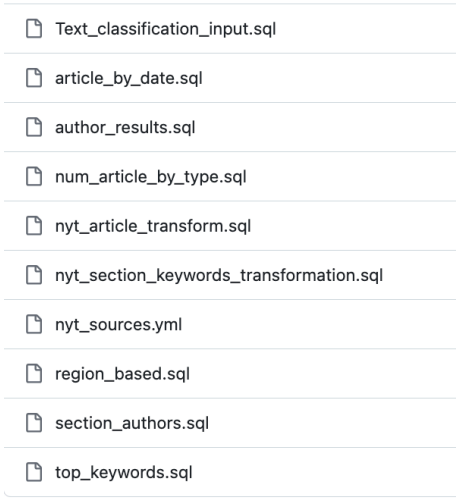


2. *A new database **NYT_DB** is created at Snowflake. A **FiveTran** partner connect is established and an initial sync of RDS is done to load the initial archive onto Snowflake's NYT_DB.*

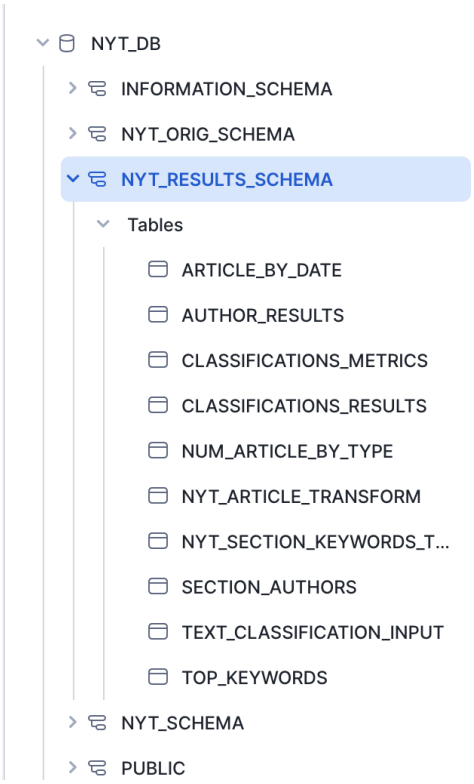


3. *A **DBT** partner connect is established to create tables and views that would cater to analysis and business decisions. (An instance of ELT)*

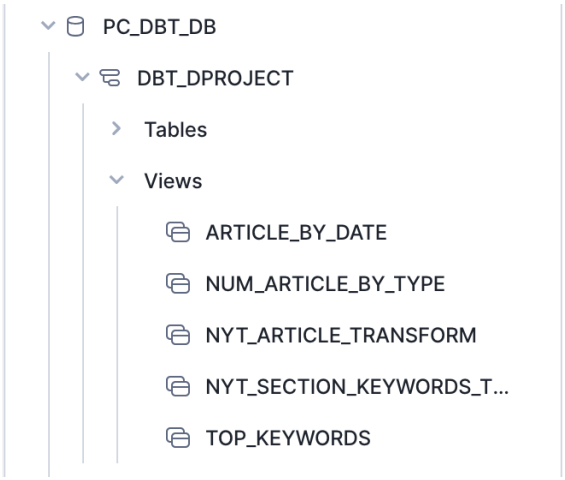




Tables created are at NYT_RESULTS_SCHEMA:



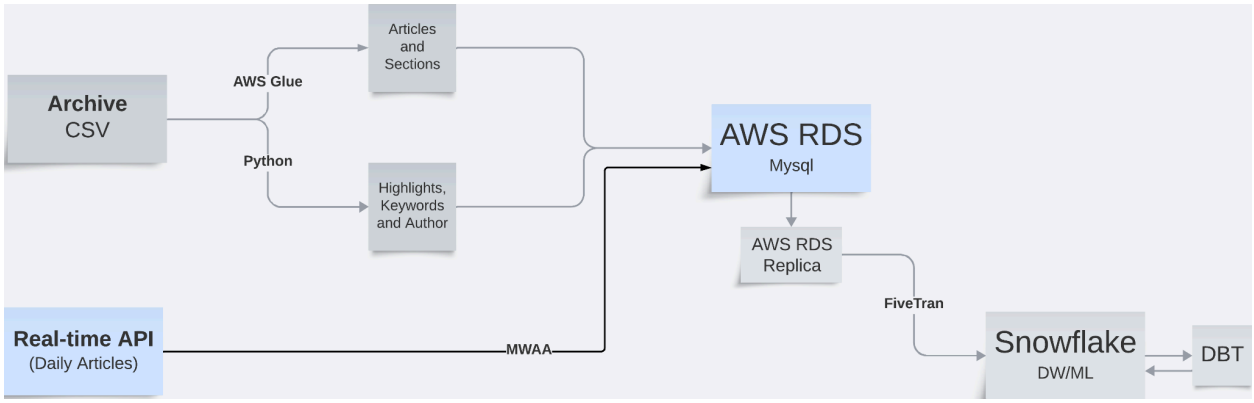
Views created within PC_DBT_DB:



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

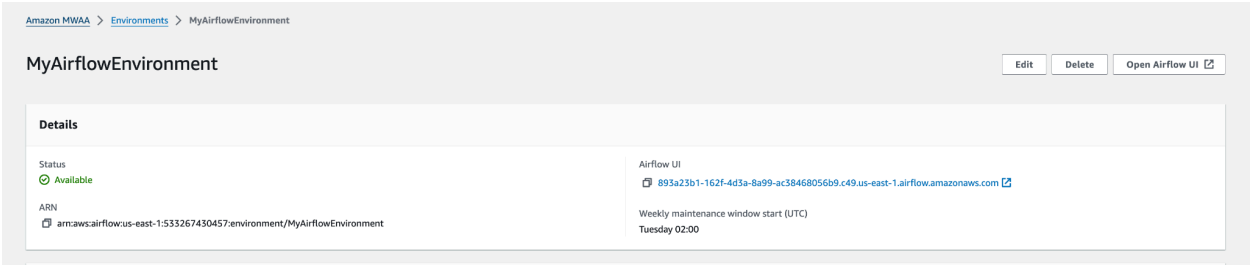
The **Daily Articles API** is extracted, transformed into a compatible schema and loaded into the same **RDS** instance. This serves as the operational database, which also eventually gets synced to **Snowflake**. This ETL from API to RDS is orchestrated by Amazon's **MWAA** and the DAG runs once everyday.

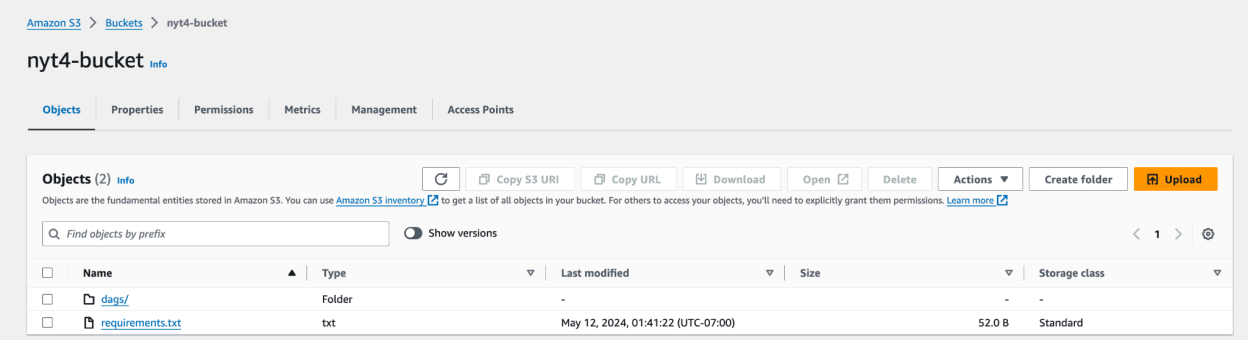


a. **Daily Articles API to be used :**
https://developer.nytimes.com/docs/articlesearch-product/1/routes/articlesearch.json/get
The API response has a list of articles on the particular day that resembles the metadata of the Archive data from the NY Times dataset. This API will be used to add the daily articles data to the RDS which already contains data from the NY Times Archive dataset.

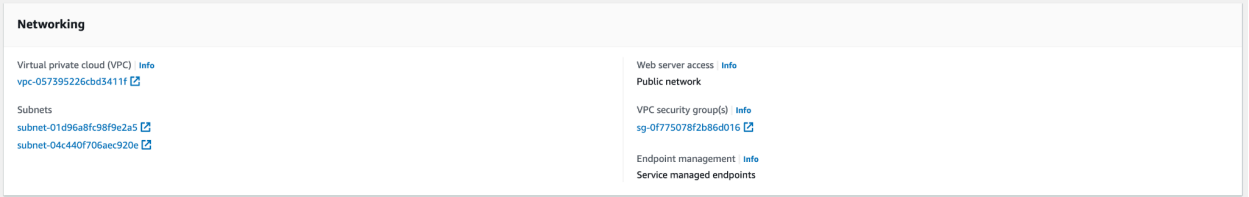
The ETL to fetch Daily Articles from the API, make it compatible with the RDS schema and then to append to RDS MySQL DB is orchestrated by Managed Apache Airflow (MWAA).

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





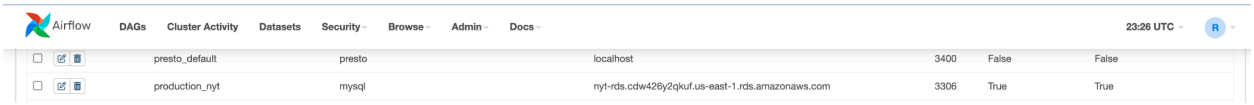
A VPC with two subnets is created.



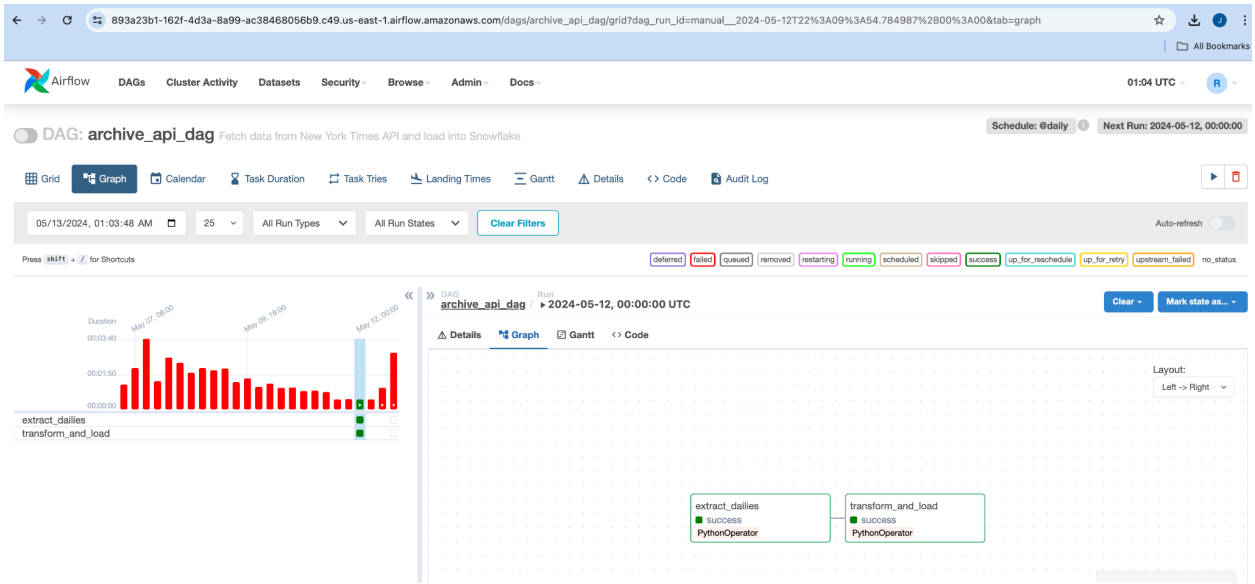
c. DAG

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:



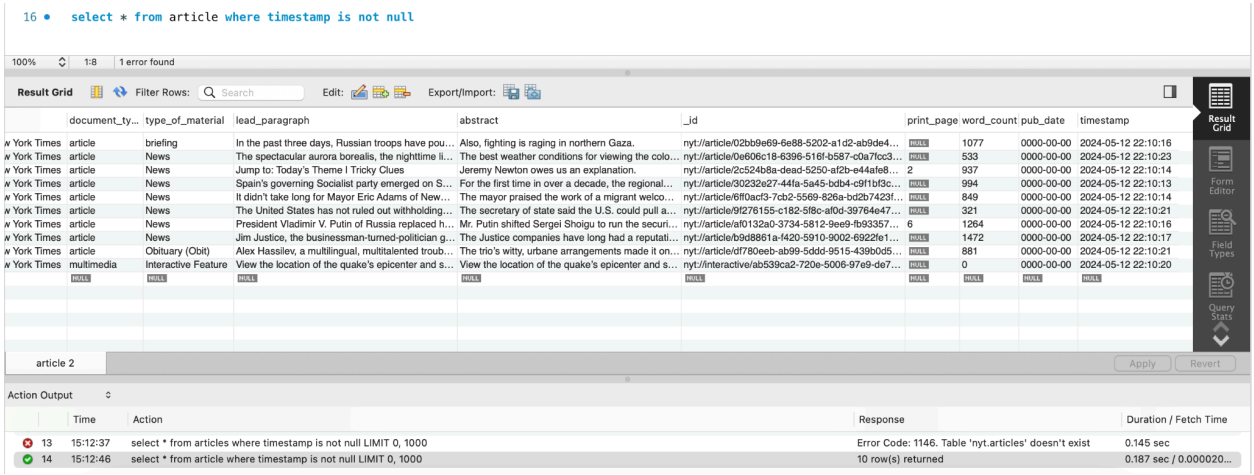
The DAG is configured to run once everyday to fetch the day's articles.
Example of a run:



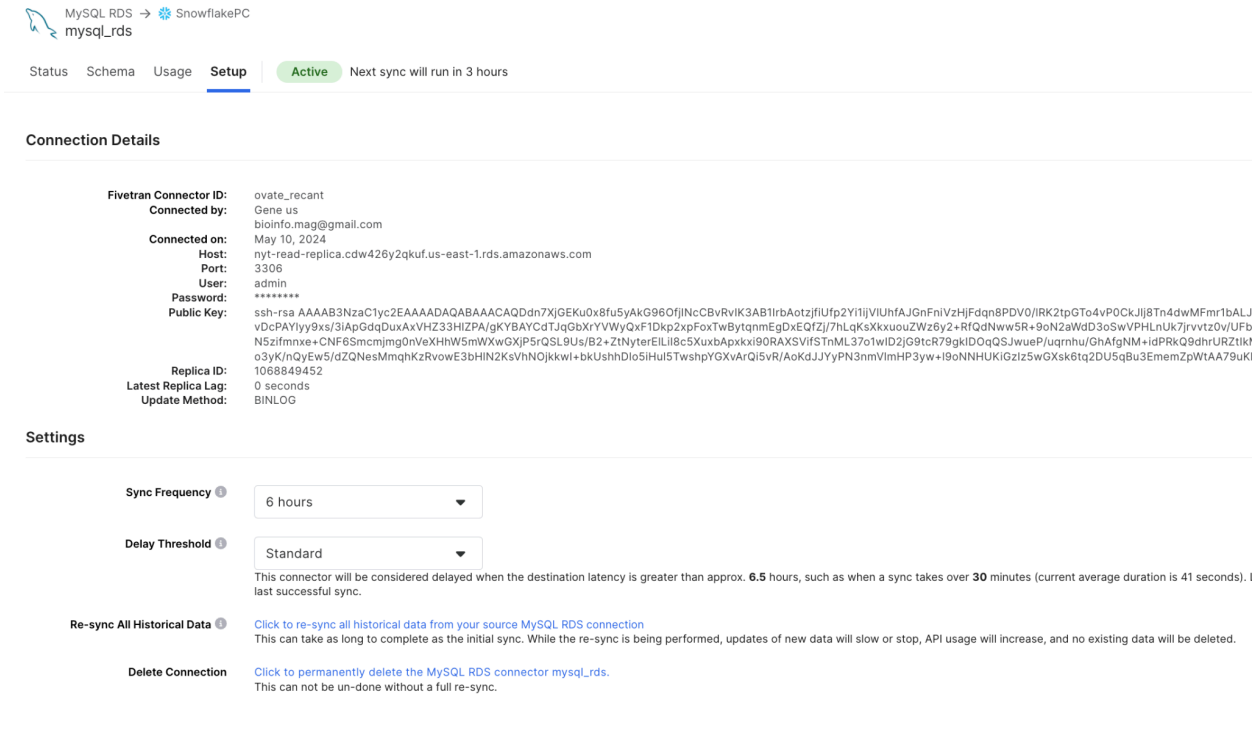
Logs:

```
[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:201}} 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
```

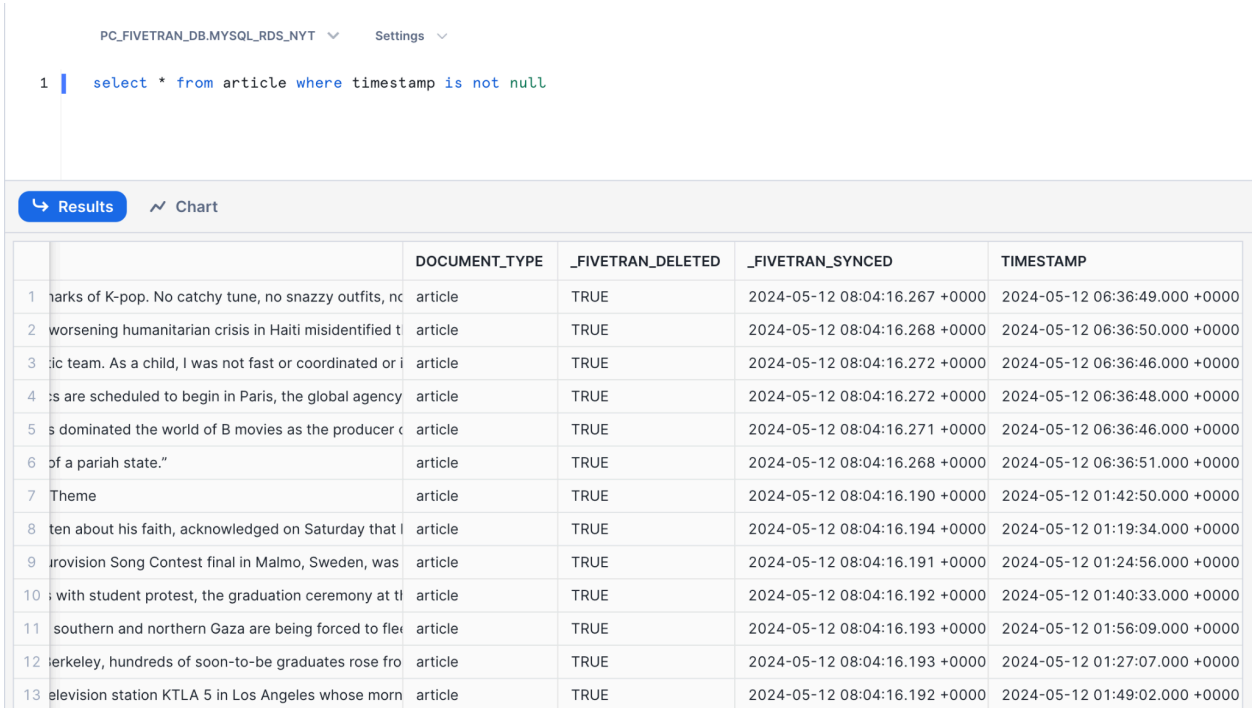
After DAG run, Daily Articles data accessed from RDS:



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



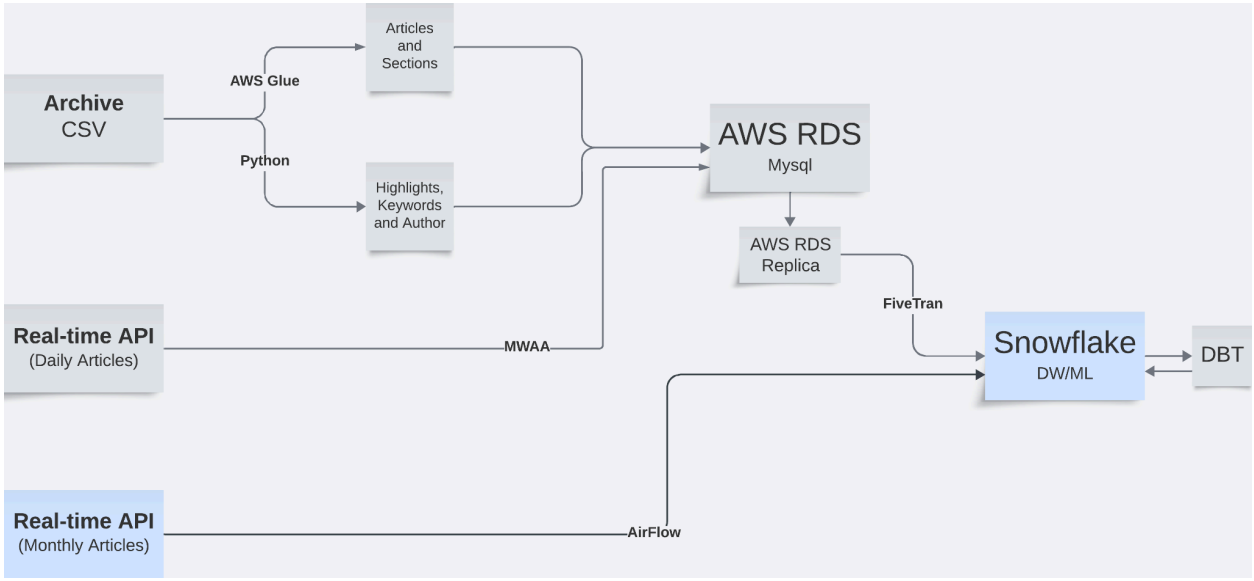
Data at Snowflake after eventual sync:



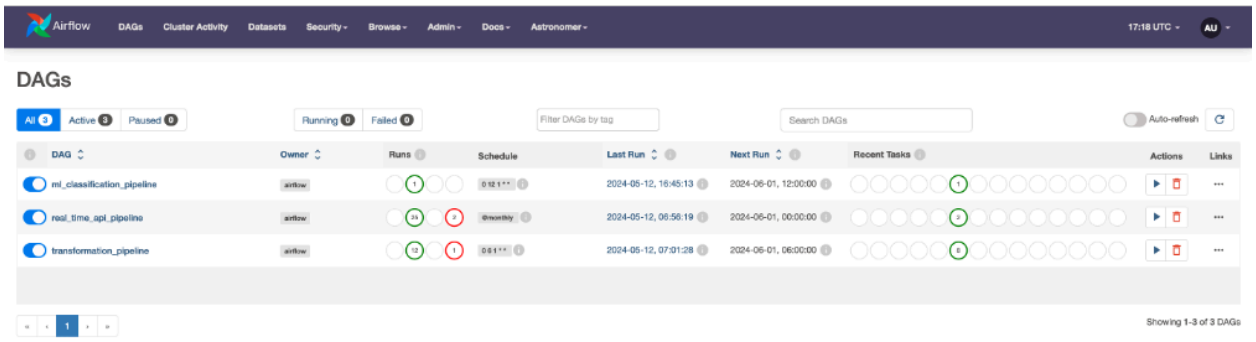
		DOCUMENT_TYPE	_FIVETRAN_DELETED	_FIVETRAN_SYNCED	TIMESTAMP
1	harks of K-pop. No catchy tune, no snazzy outfits, nc	article	TRUE	2024-05-12 08:04:16.267 +0000	2024-05-12 06:36:49.000 +0000
2	worsening humanitarian crisis in Haiti misidentified t	article	TRUE	2024-05-12 08:04:16.268 +0000	2024-05-12 06:36:50.000 +0000
3	ic 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	s 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 c	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	rovision 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 tl	article	TRUE	2024-05-12 08:04:16.192 +0000	2024-05-12 01:40:33.000 +0000
11	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	erkeley, 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

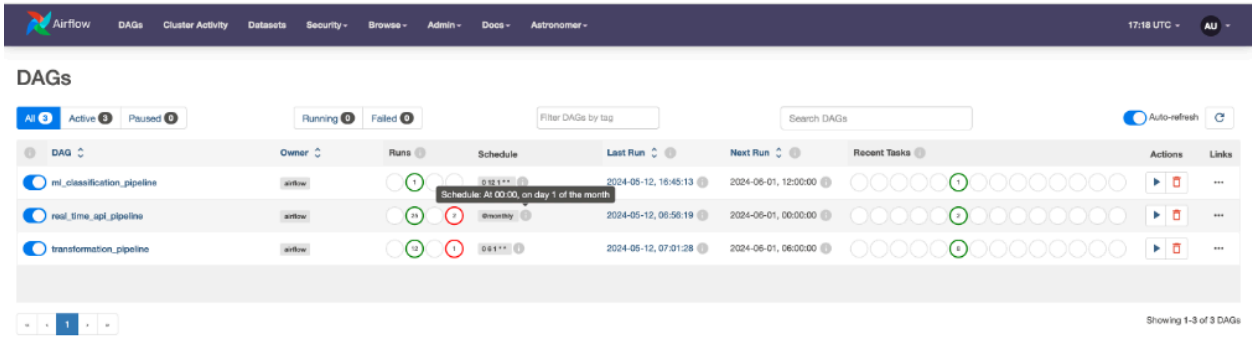


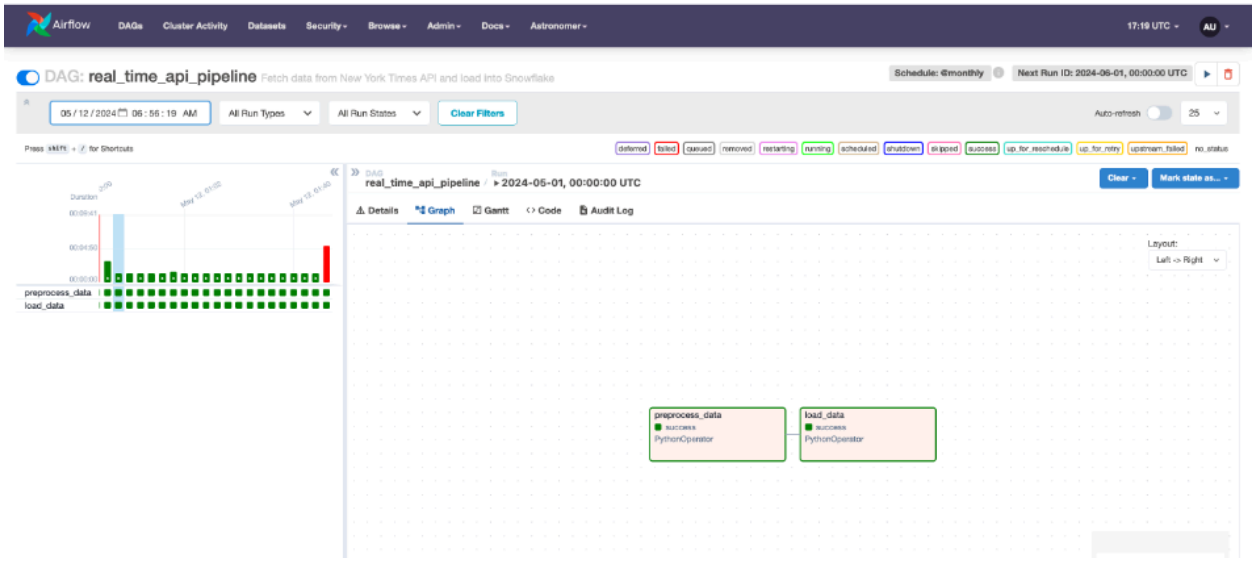
The API returns an array of NYT articles for a given month and year. Its response fields are the same as the Article Search API which is loaded on a daily basis. Updating snowflake warehouse with data on a monthly basis on top of the data from Article Search API will ensure no article data is missed 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.



Real Time API Processing pipeline:

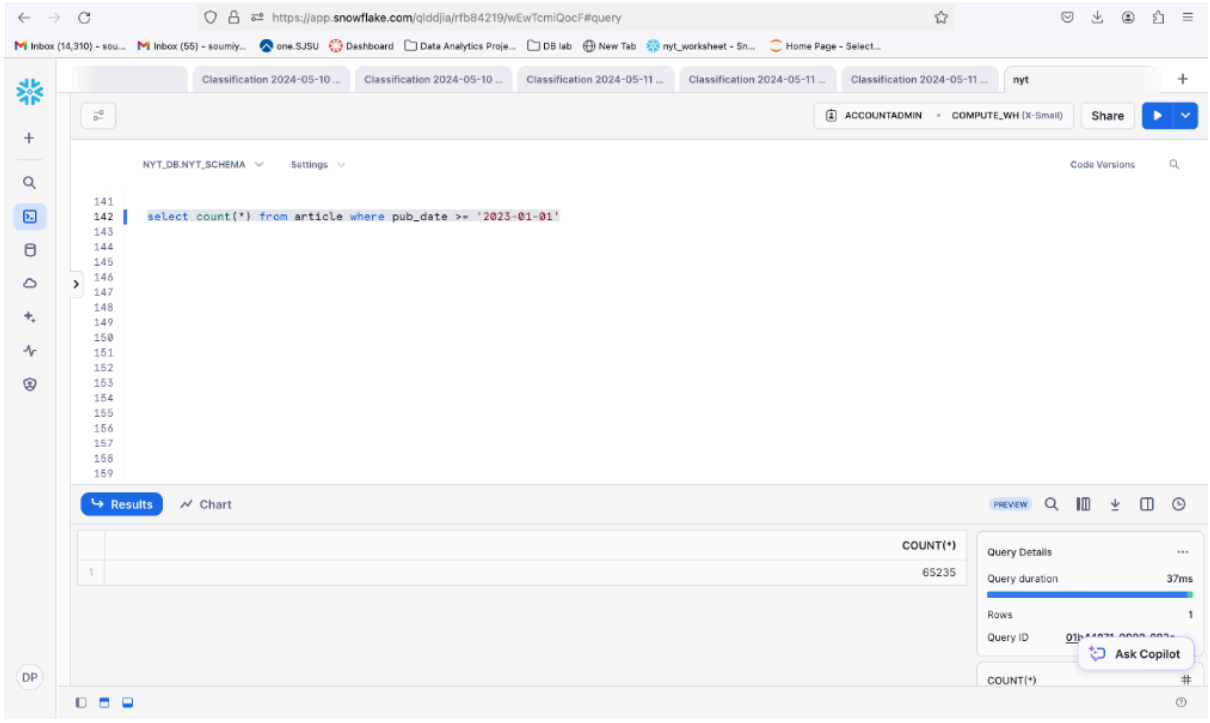
A DAG pipeline with python operators fetches the previous month articles data from NYT Archive API, processes and loads it to Snowflake warehouse if there are any new articles found. During the data processing step the data is extracted and formatted into tables to match the schema of Snowflake Warehouse. 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.





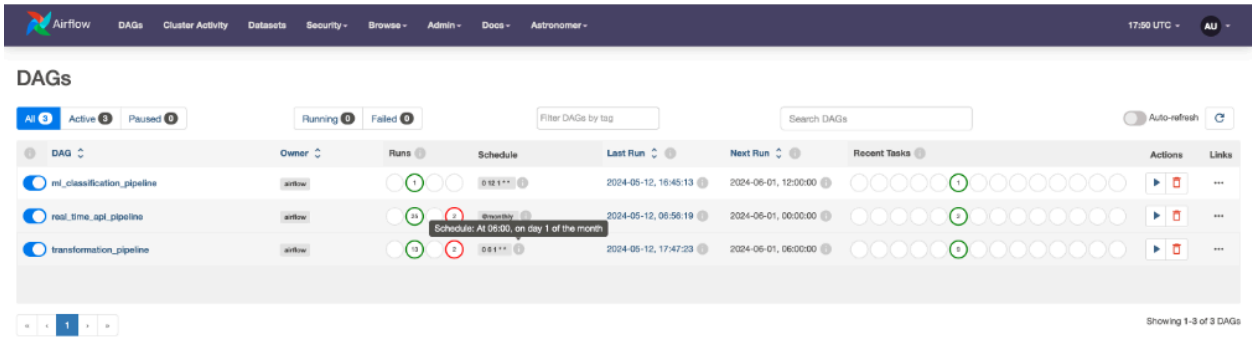
Also, this pipeline was manually triggered to load some of the historic data missing in the database. The number of articles inserted into the Snowflake database NYT_DB.NYT_SCHEMA from January 2023 to

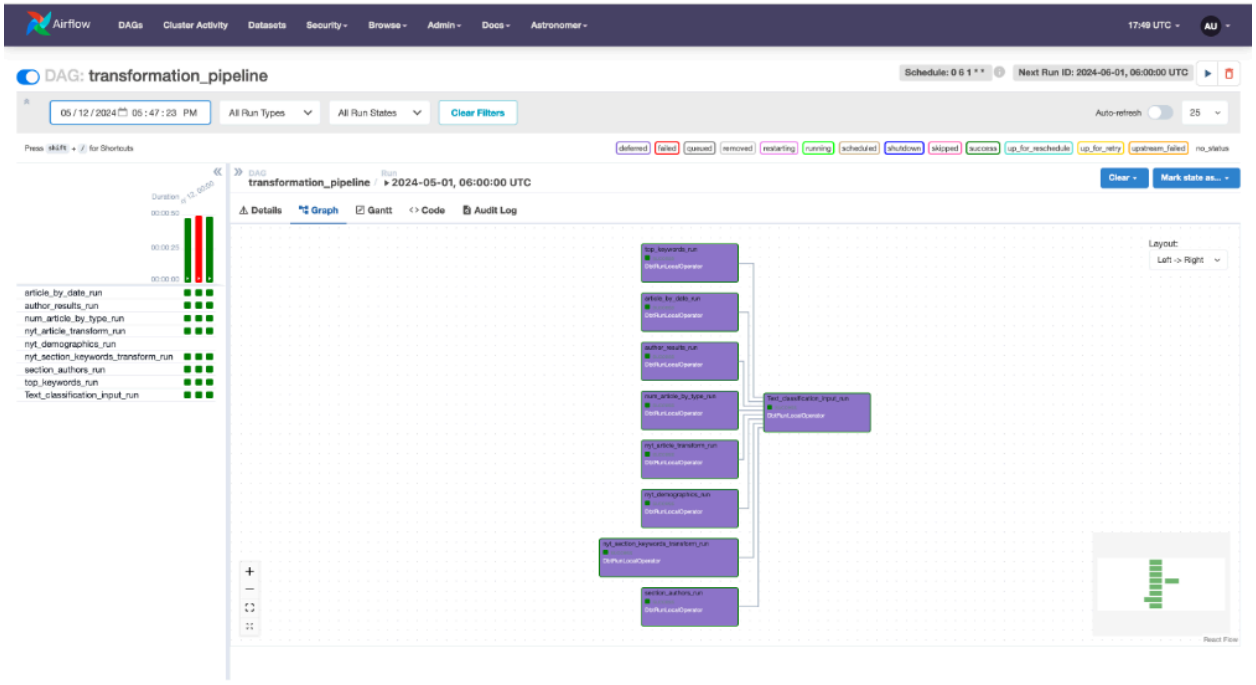
April 2024 by calling the API to update the monthly articles data is shown in the query result below:



Transformation Pipeline:

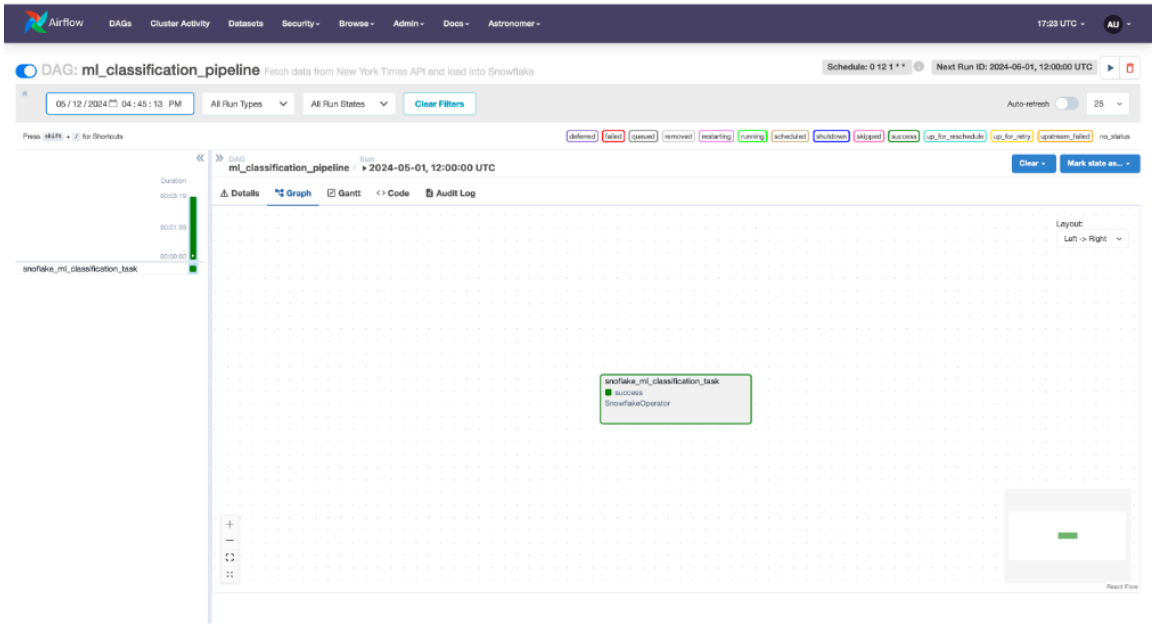
A DBTDAG pipeline performs dbt transformations and generates summarized result tables for analytics, visualizations and ML analysis. The result tables are stored in the Snowflake database NYT_DB.NYT_RESULTS_SCHEMA. This pipeline is scheduled to run every 1st day of the month at 6 am. This is scheduled 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.





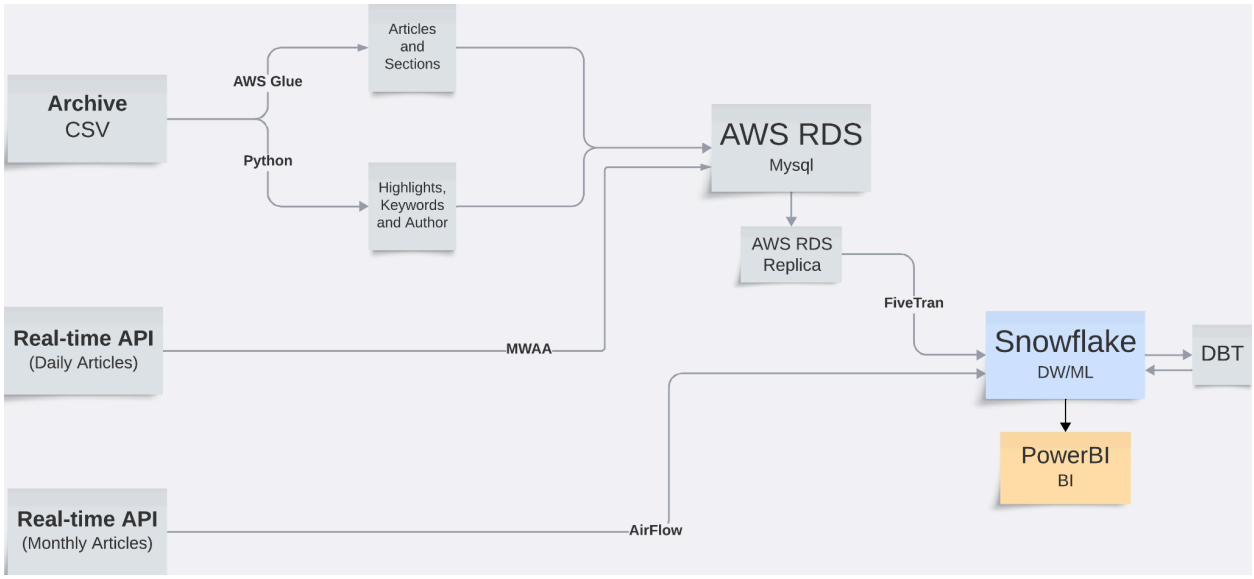
Machine Learning Text Classification Pipeline:

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:

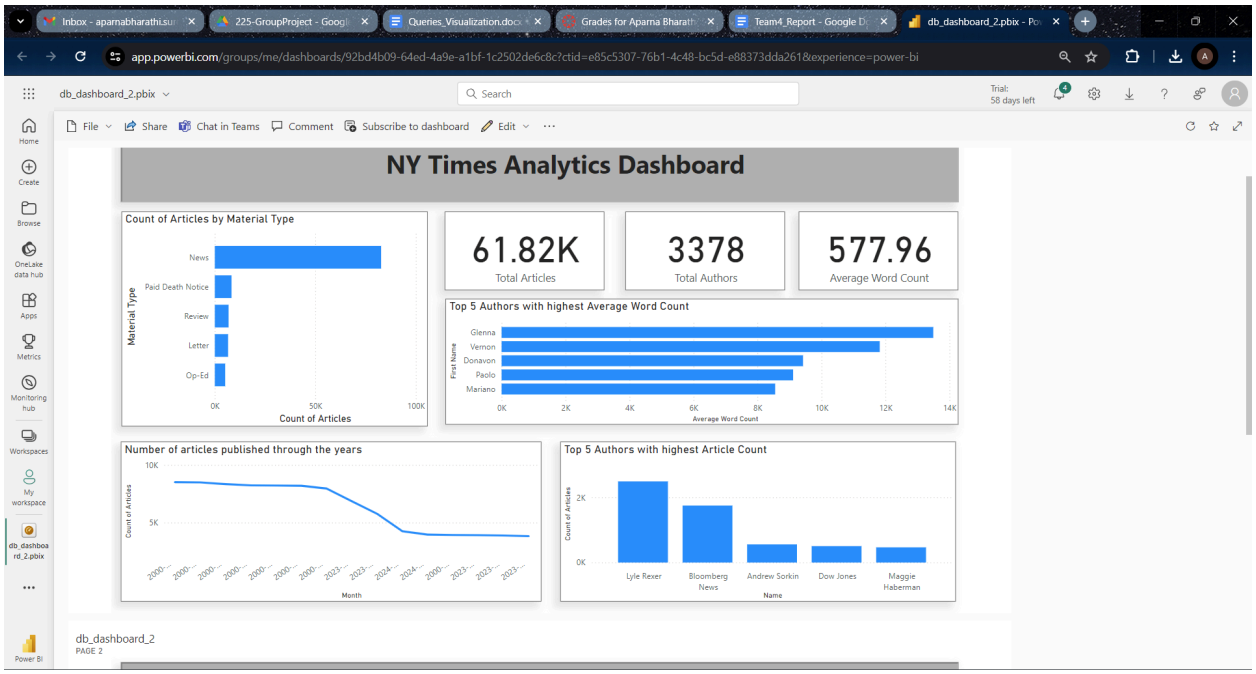
1. Open Power BI Desktop and click "Get Data" from the Home ribbon.
2. In the "Database" category, select "Snowflake" and then "Connect".
3. Enter the Snowflake server address and warehouse name.
4. Provide your Snowflake username and password and click "Sign in".
5. Choose the tables or views you want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

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

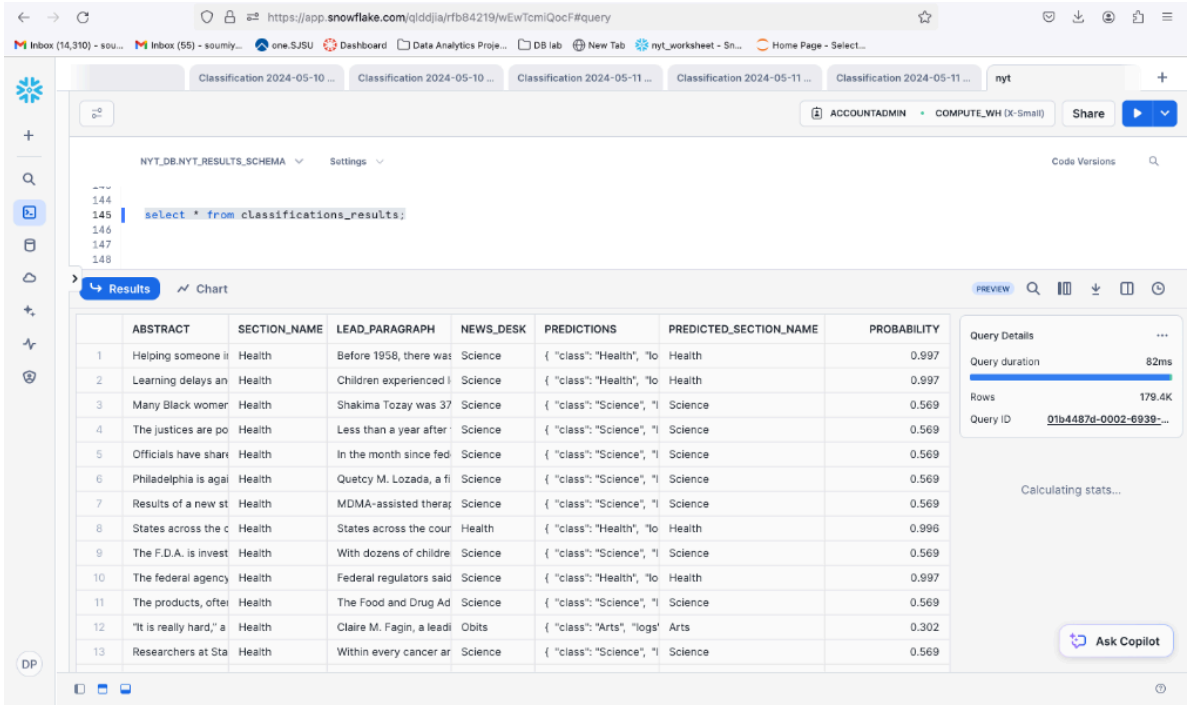
Please find the queries used for statistical analysis here.
Please find the Corresponding DBT Models here
Please find the visualization screenshots here.

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.

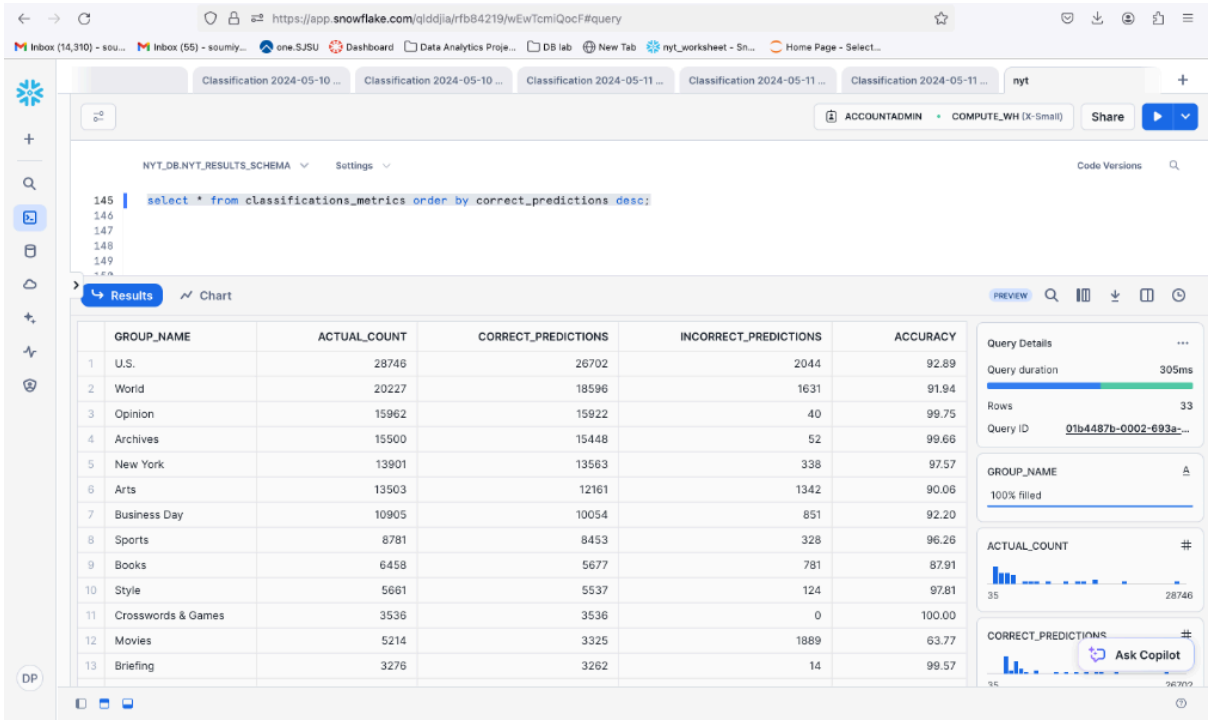
The process involves feeding the abstract (a summary), the lead paragraph (the opening paragraph), and information about the news desk (the department or team within a publication responsible for certain topics) of an article into a model. The model then uses this information to automatically categorize 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:



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.

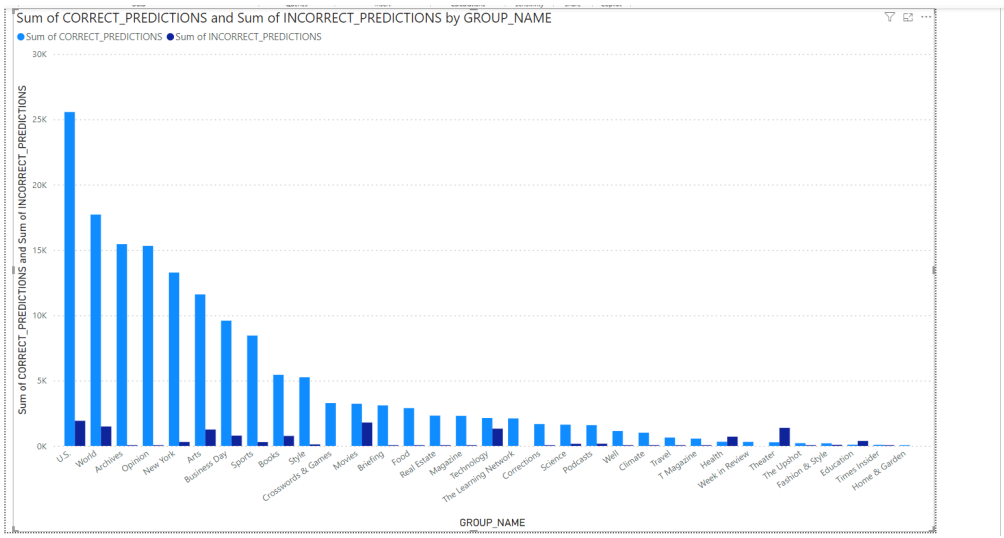
Table showing the classification metrics for section name prediction:



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

	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

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