PolitiCast: An Election Prediction System

Karthik Shetty Nagaraja[#1] ,Durga Samhitha Muvva[#2], Nandana Chigaterappa HemanthKumar[#3], Nithin Keshavamurthy[#4], Varun Reddy Bhumi Reddy[#5] , Yashaswini Chowdary Madineni[#6]

[#]*Data Analytics, San José State University*
1 Washington Sq, San Jose, CA 95192

[1]karthikshetty.nagaraja@sjsu.edu
[2]samhitha.muvva@sjsu.edu
[3]Nandana.chigaterappahemanthkumar@sjsu.edu
[4]nithin.keshavamurthy@sjsu.edu
[5]varunreddy.bhumireddy@sjsu.edu
[6]yashaswini.madieni@sjsu.edu

***Abstract*— Database Management Systems (DBMS) facilitates data storage, retrieval, and manipulation, which are critical components for analytics. This project focuses on developing an Election Prediction System that leverages DBMS capabilities to analyze political articles from the "NYT Articles: 2.1M+ (2000-Present) Daily Updated" dataset and forecast election outcomes. The system utilizes historical and real-time data, including demographics, socioeconomic indicators, previous election results, and sentiment data extracted from the articles, to predict the potential success of political candidates or parties. Relevant information is filtered using keywords, while unnecessary columns are removed during data preprocessing. The cloud-based database ensures scalability, accessibility, and data security. The system's accuracy and reliability in predicting election results are evaluated through testing and validation against election data. Visualization techniques are employed to derive insights from the dataset, enabling improved decision-making for political stakeholders, campaign strategists, and voters.**

Keywords*— DBMS, Election Prediction System, New York Times, Results, Sentiment, real-time

I. Problem Statement

In the political landscape, predicting the election results accurately is crucial for both voters and political parties. The New York Times Articles dataset extracted from Kaggle, containing over 2.1 million articles from 2000 to the present, including data relevant to socioeconomic indicators and previous election results. The major challenge is to effectively utilize this vast dataset to build a robust Election Prediction System that can make use of the tools existing in the market. While the archived data in the NYT dataset offers valuable insights, it is only limited to March, 2000, which demands for incorporation of real-time data to enhance accuracy of prediction and development of a comprehensive project pipeline that can handle both archived and real-time data accurately.

The primary objective of the project is to design and implement a scalable pipeline that can extract, process, and analyze the NYT dataset incorporating real-time data extracted from API. The pipeline should use appropriate data tools and technologies to ensure data integrity, governance, and security throughout the process. Furthermore, the pipeline should include advanced techniques such as Natural Language Processing (NLP) to extract relevant features from textual data, enabling accurate sentiment analysis and topic modeling. Machine learning models trained on this processed data should be capable of predicting election outcomes with high accuracy, considering various factors.

II. Dataset Selection & Importance

The New York Times, which provides in-depth reporting on a variety of subjects, such as politics, economics, social issues, and more is used for PolitiCast. The dataset is downloaded from Kaggle, which includes a variety of historical data from 2000 to the present, in this case is required spotting long-term trends and patterns in news coverage and public opinion. The project specially used the politics related articles as it deals with election prediction. The data has been filtered and preprocessed using the same criteria.

A. Archival Data

- Source: Kaggle (New York Times Headlines - Archived Data)
- Link: https://www.kaggle.com/datasets/aryansingh0909/nyt-articles-21m-2000-present

The archival data downloaded from the source is preprocessed and narrowed down to a manageable size of around 250 MB. Our analysis of patterns over the last 2 decades is made possible by historical data, which offers context and depth. grasp how public opinion and media coverage have changed throughout time requires a grasp of this.

B. Real-Time Data

- Source: New York Times Archive API
- Link: https://developer.nytimes.com/apis

The Archive API regularly updates the dataset with new items, ensuring our analysis stays current and incorporates the most recent news. The integration of both real-time and historical data enables quick analysis and ongoing monitoring, essential for tracking the impact of current events and public sentiment and responding promptly to news updates. Archive API is used to extract the real time data, as the column structure matches with the dataset on Kaggle, there by facilitating the incremental load task.

Both datasets (archival and real-time) have matching columns, which simplifies the ETL process and ensures consistency in our analysis.

The following are the descriptions of the columns in the dataset:

1. abstract: A brief summary of the article.
2. web_url: The URL of the article on the New York Times website.
3. snippet: A snippet or excerpt from the article.
4. lead_paragraph: The opening paragraph of the article.
5. print_section: The section of the newspaper where the article appeared (if available).
6. print_page: The page number of the newspaper where the article appeared (if available).
7. source: The source of the article, typically "The New York Times".
8. multimedia: Multimedia elements associated with the article, such as images or videos.
9. headline: The main headline of the article.
10. keywords: Keywords associated with the article, indicating important topics or entities covered.
11. pub_date: The publication date of the article.
12. document_type: The type of document (e.g., article, op-ed, etc.).
13. news_desk: The news desk responsible for the article.
14. section_name: The section name of the article within the newspaper.
15. subsection_name: The subsection name within the section.
16. byline: The author(s) of the article.
17. type_of_material: The type of material (e.g., news, editorial, etc.).
18. _id: The unique identifier for the article.
19. word_count: The word count of the article.
20. uri: The uniform resource identifier for the article.

This extensive dataset offers a wealth of data, facilitating in-depth examination of political news coverage throughout time. We can monitor shifts in media attitude and coverage by utilizing both real-time and archived data, providing insightful information about the political environment. The data after performing incremental load is used for visualization and statistical analysis.

III. Database Configuration

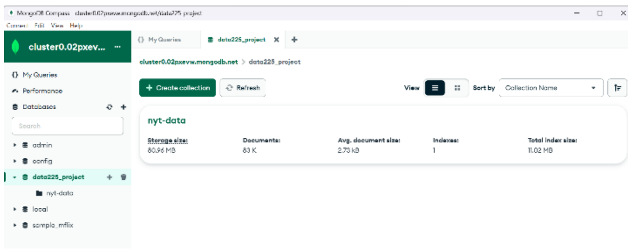
A. Pre-Processing

The project deals with 2 files: archived data and real-time data, two different approaches are used to deal with these data. The archived data when downloaded was around 4GB. The major idea is to filter data that is related to politics, as the system performs election prediction. The data is filtered out based on subsection name: politics. The null rows are dropped, and the pre-processing narrowed down data t0 250MB.

The real-time data on the hand, was fetched using a MWAA DAG task month by month in JSON format. The same task combines the files into single JSON file and converts it to CSV to make the Incremental load easier,

B. *Database Setup*

In order to create the data in JSON format, MONGO is used as it has the capability to create nested documents out of array and dictionary structures. Mongo Atlas and Compass are used for the database setup. Pymongo library is used, which connects the python environment to mongo cluster using a connection string. Python code is employed to push the csv data into cluster to achieve JSON format (with proper nested document structure). The JSON data is stored in a database named 'data225_project' under schema 'nyt-data'. The following is the connection string:
'mongodb+srv://samhithamuvva:Salmonskinroll%406
@cluster0.02pxevw.mongodb.net/'



nyt-data

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
80.96 MB	83 K	2.73 kB	1	11.02 MB

Fig 1. Mongo Compass Set-up

IV. CLOUD ARCHITECTURE

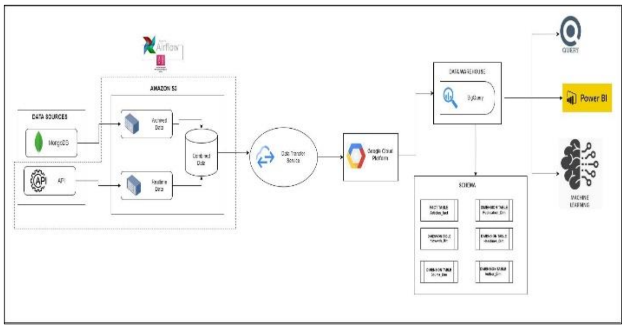


Fig 2. Cloud Architecture that implements "PolitiCast'

The architecture outlined in Figure 1 reflects a sophisticated cloud-based solution that we have tailored to handle the specific requirements of our project, 'PolitiCast', which focuses on integrating archived data with real-time data for comprehensive analytics of the Political data embedded in the New York Times articles. Below is a detailed explanation of each component, emphasizing their strategic selection based on our project's unique demands.

A. *Data Sources:*

1) Archived Data: As a NoSQL database, MongoDB is adept at managing semi-structured data. With respect to our particular use case and for our project, it was selected because of its efficiency in handling nested structures like 'headlines', 'keywords', and 'byline' that were present in our dataset. Thus, we utilized MongoDB to store our Archived Data.

2) Real-time Data: Our real-time data is fetched from the 'Archive API' of New York Times Articles and is stored in an S3 Bucket.

B. *Data Aggregation and Storage:*

1) Amazon S3: We have utilized Amazon S3 as the primary cloud storage solution due to its high durability and availability, key for safeguarding our vast data repository. It is used to store the archived data from MongoDB, the real-time data streamed through the 'Archive API' of New York Times and the combined data that we obtain through the implementation of Incremental Loading.

C. *Data Orchestration:*

1) Managed Workflows for Apache Airflow (MWAA): We have used the AWS MWAA to orchestrate our complex workflows, particularly to manage our ETL process that transforms raw data into a format suitable for analysis. Airflow's programmable environment allows for customizing pipelines that automate and synchronize data tasks across different cloud services, which is crucial for maintaining a reliable data flow and timely updates.

D. *Data Transfer:*

1) Data Transfer Service: We have used Google Cloud Platform's BigQuery Data Transfer Service to ensure a seamless and secure data transfer from our AWS S3 bucket to Google BigQuery based on a particular frequency (once every 24 hours), which is optimized for minimal latency in data availability. Its efficiency is vital for maintaining the integrity and timeliness of the data fed into BigQuery, directly impacting the responsiveness of our analytical outputs.

E. Data Processing and Warehousing:

1) Google BigQuery: We have chosen Google BigQuery as our Data Warehouse solution as it offers a more integrated experience for semi-structured data thanks to its native support for JSON and other flexible data types within SQL queries. Also, due to its serverless architecture, it eliminates the need for manual scaling and resource management. This is especially beneficial for handling fluctuating data volumes and query loads, ensuring cost-effective data processing. Its integration with Google Cloud services enhances workflow efficiency and simplifies the analytics pipeline, while support for JSON and other semi-structured data types facilitates straightforward querying of our complex data.

F. Data Analysis and Reporting:

1) Power BI: For visualization and reporting, we have utilized Power BI which supports BigQuery as a Data Source. We have converted our complex BigQuery analytical results into comprehensible visual reports through PowerBI. Its rich graphical capabilities allow us to interact with the data, enhancing understanding and facilitating informed decisions based on the latest analytics.

V. IMPLEMENTATION

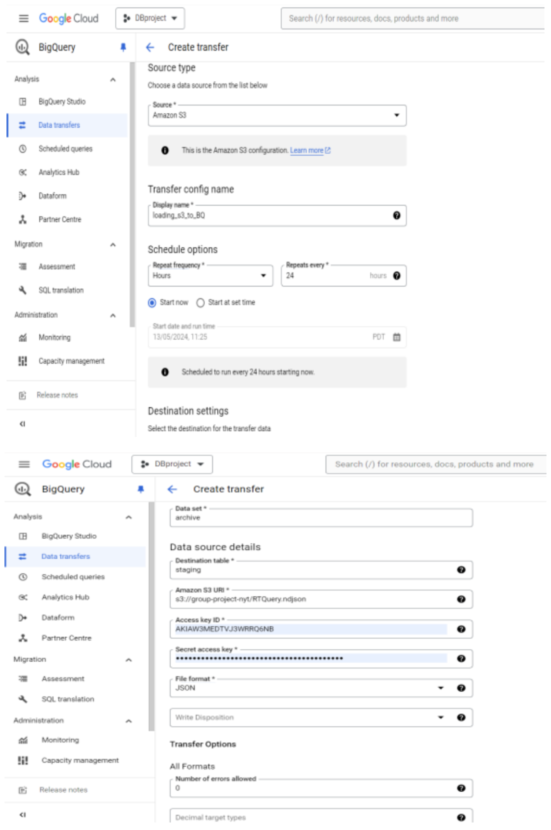
A. ETL Pipeline

The ETL (Extract, Transform, Load) process is orchestrated using Apache Airflow, which provides robust scheduling and monitoring capabilities essential for managing our data workflows efficiently. The project uses ETL (Extract, Transform, Load) over ELT (Extract, Load, Transform) because we are utilizing JSON for querying. ETL allows us to preprocess and transform the data before loading it into the database, optimizing both performance and compatibility with JSON querying. Additionally, ETL facilitates complex data integration and ensures data quality and consistency through preprocessing steps, aligning well with the project querying and data analysis.

1) Extract: Data is sourced from two main sources: archived data and real-time API data. Apache Airflow automates the extraction process, ensuring that data from MongoDB and API is consistently and reliably captured. This step is crucial for acquiring the full spectrum of data required for our analysis. The archived data from Mongo is placed in an S3 bucket. A DAG script is written to fetch the data from API. The data is fetched month by month, is combined into one single json file that is converted to csv and pushed into S3 bucket. These two files are used further for transformation and performing queries.

2) Transform: The transformation of data is performed on the file not just at one point. A DAG task is written that combines archive data with realtime data based on pub_date column, that has time stamp. New rows are imported if there is new entry in pub_date column. For upload function, the code takes pub_date column and if the value is same, it does comparison based on _id and if there is match it will take the latest one and upload it to the combined data. The final file after performing the transformation is named as combined_result, which is pushed into BigQuery. Furthermore, DAG scripts are written for file formatting to convert data into a structure that is compatible with our data warehouse in BigQuery. This step is vital for cleaning and preparing the data, which includes handling anomalies, formatting dates, and structuring nested JSON fields into a schema that can be effectively queried.

3) Load: The final step involves pushing the transformed data into Google BigQuery. BigQuery has been chosen for its ability to handle large volumes of semi structures data and is well compatible for JSON format. BigQuery's seamless integration with other Google Cloud services allows us to further enhance our data analytics processes, from simple queries to complex machine learning models. Data Transfer Service, which is part of Google Cloud Platform is used to push the data from S3 bucket to Big Query which is further used for querying. The Data Transfer Service requests us to fill in certain details to establish connection between AWS S3 bucket and BigQuery Data.



Create transfer

Source type

Choose a data source from the list below

Source *: Amazon S3

This is the Amazon S3 configuration. Learn more

Transfer config name

Display name *: loading_s3_to_BQ

Schedule options

Repeat frequency *: Hours

Repeats every *: 24 hours

Start now [✓] / Start at set time

Start date and run time: 13/05/2024, 11:25 PDT

Scheduled to run every 24 hours starting now.

Destination settings

Select the destination for the transfer data

Create transfer

Data set *: archive

Data source details

Destination table *: staging

Amazon S3 URI *: s3://group-project-nyt/RTQuery.ndjson

Access key ID *: AKIAW3MEDTVJ3WRRQ6NB

Secret access key *: ••

File format *: JSON

Write Disposition

Transfer Options

All Formats

Number of errors allowed: 0

Decimal target types

Fig 3. Data Transfer Service Connection: S3 to Big Query

B. MWAA Setup

The project uses MWAA for orchestrating the ETL Pipeline. The steps involved in the Configuration of Apache Airflow on Amazon Managed Workflows for Apache Airflow (MWAA) are listed below:

1) Introduction: This section describes the comprehensive setup of Apache Airflow within the AWS ecosystem, tailored to meet the specific needs of our project which involves orchestrating complex ETL tasks for Data Analysis. The configuration details provided ensure a robust, scalable, and secure environment capable of handling our data workflows efficiently.

2) AWS Account Setup: We confirmed administrative access to crucial AWS services, including MWAA, S3, IAM, and VPC. This level of access is imperative to manage resources effectively and with the necessary oversight, ensuring full control over resource creation, modification, and management across the project lifecycle.

3) S3 Bucket Initialization:

<u>Creation:</u>

We initialized an S3 bucket named `group-project-nyt` dedicated to storing our workflow scripts and requisite libraries. The bucket acts as the central repository for all Airflow-related files and is critical for the proper functioning of our data workflows.

<u>DAG Uploads:</u>

We then uploaded the essential DAG files designed to orchestrate our data workflows directly to the newly created S3 bucket. This setup step is crucial for kick-starting our automated data processing tasks within the Airflow environment.

4) Security and Access:

<u>IAM Role Configuration:</u>

We then crafted a custom IAM policy, `MWAAExecutionPolicy`, granting essential privileges for accessing and interacting with the `group-project-nyt` S3 bucket, CloudWatch Logs, KMS, MWAA, and SQS.

The policy grants the following permissions:

- **CloudWatch:** Limited Write access to all resources.

- **CloudWatch Logs:** Read, List, and Write access to multiple resources.

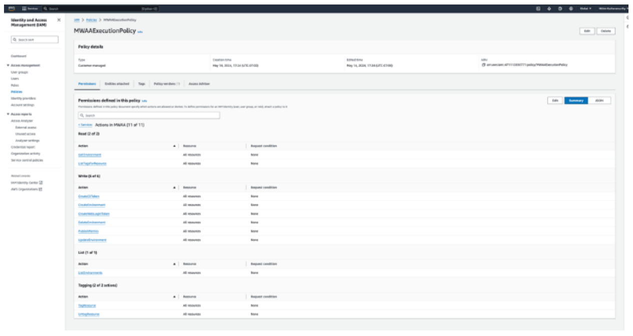
- **KMS:** Read and Write access.

- **MWAA:** Limited Write access to resources in the `us-east-1` region.

- **S3:** Read and List access to multiple resources.

- **SQS:** Read and Write access to specific queues and regions.

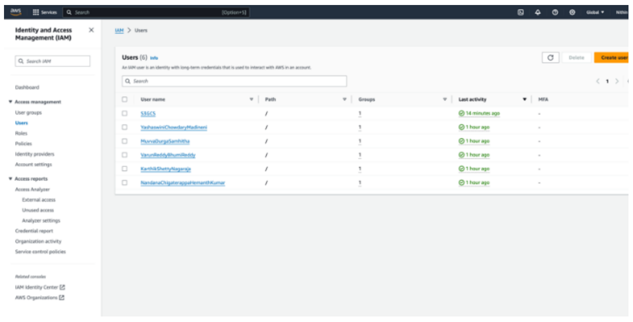
Fig 4 indicates the policy details specified for IAM policy creation.



MWAAExecutionPolicy

Fig 4. IAM Policy Details

Along with this, the project has also setup access for all users who are working on this project. Fig 5 shows the different IAM users accessing the project.



Identity and Access Management (IAM)

IAM > Users

Users (6)

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

User name	Path	Groups	Last activity	MFA
S3GCS	/	1	14 minutes ago	-
YashaswiniChowdaryMadineni	/	1	1 hour ago	-
MuvvaDurgaSamhitha	/	1	1 hour ago	-
VarunReddyBhumiReddy	/	1	1 hour ago	-
KarthikShettyNagaraja	/	1	1 hour ago	-
NandanaChigaterappaHemanthKumar	/	1	1 hour ago	-

Fig 5: IAM Users

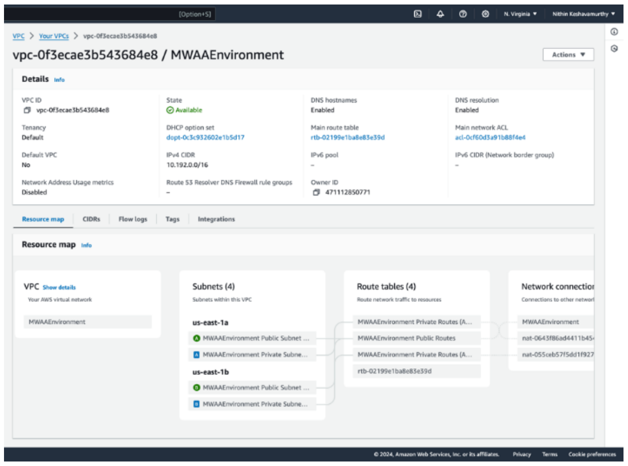
5) Network Setup:

<u>Virtual Private Cloud (VPC):</u>

We veveraged an existing VPC named `MWAAEnvironment` configured with private subnets to ensure secure and isolated network traffic. The VPC setup includes:

- **Subnets**: Configured with both public and private subnets across availability zones `us-east-1a` and `us-east-1b`.

- **Route Tables**: Appropriate routing configured to ensure secure and efficient data flow between subnets. Fig 6 represents the VPC details in AWS.



vpc-0f3ecae3b543684e8 / MWAAEnvironment

Details

VPC ID: vpc-0f3ecae3b543684e8	State: Available	DNS hostnames: Enabled	DNS resolution: Enabled
Tenancy: Default	DHCP option set: dopt-0c3c932602e1b5d17	Main route table: rtb-02199e1ba8e83e39d	Main network ACL: acl-0cf60d3a91b88f4e4
Default VPC: No	IPv4 CIDR: 10.192.0.0/16	IPv6 pool: –	IPv6 CIDR (Network border group): –
Network Address Usage metrics: Disabled	Route 53 Resolver DNS Firewall rule groups: –	Owner ID: 471112850771	

Resource map

VPC: MWAAEnvironment

Subnets (4): us-east-1a: MWAAEnvironment Public Subnet ..., MWAAEnvironment Private Subne...; us-east-1b: MWAAEnvironment Public Subnet ..., MWAAEnvironment Private Subne...

Route tables (4): MWAAEnvironment Private Routes (A..., MWAAEnvironment Public Routes, MWAAEnvironment Private Routes (A..., rtb-02199e1ba8e83e39d

Network connections: MWAAEnvironment, nat-0643f86ad4411b45, nat-055ceb57f5dd1f927

© 2024, Amazon Web Services, Inc. or its affiliates.

Fig 6: VPC details AWS

This configuration is crucial for maintaining network security and performance, ensuring that Airflow operations are conducted in a controlled environment.

6) Environment Deployment:

- MWAA Environment Creation:

We configured the MWAA environment through the AWS Management Console with the following specific parameters:

- **Environment Name**: `NYTAirflowEnvironment`

- **Execution Role**: `AdministratorAccess` with attached custom `MWAAExecutionPolicy`.

- **S3 Bucket Location**: `s3://group-project-nyt`

- **Airflow Version**: Selected version `2.4.3` for its stability and robust feature set, ensuring compatibility and reliability.

- **Networking Details**: Integrated with the `MWAAEnvironment` VPC, utilizing selected private subnets coupled with an appropriate security group to maintain strict network security protocols. Airflow UI Interaction

- Access:

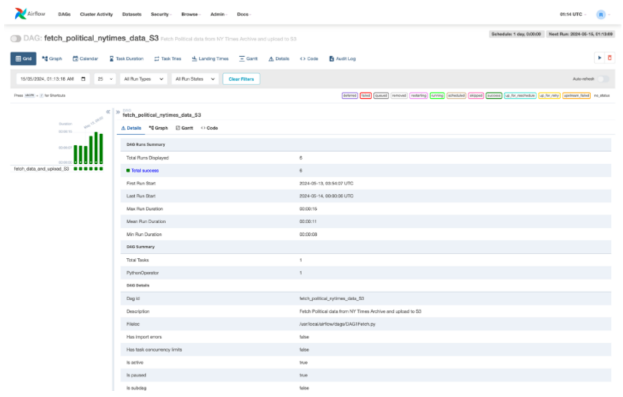
We navigated to the Airflow web interface using the URL provided in the MWAA console, enabling real-time monitoring and management of data pipelines.

- Workflow Management:

Commenced regular deployments of additional DAGs to handle evolving data processing needs, facilitating dynamic scalability and adaptability of our data handling infrastructure.

C. *MWAA TASKS:*

1) Data Fetching: CSVDAG.py (named as fetching_real-time_api_data_to_S3.py in the Code Details Document): The `CSVDAG.py` initiates the pipeline by fetching political news articles from the New York Times API corresponding to specific dates. It filters relevant fields from the JSON response and formats them into a CSV structure. These CSV files are then uploaded to the S3 bucket `group-project-nyt` under the directory structure `NYTData/year/month.csv`. This step is critical as it populates the bucket with the initial raw data set, which is essential for the incremental load and further processing by subsequent DAGs. Figure 7 shows the details and execution status of CSVDAG.py



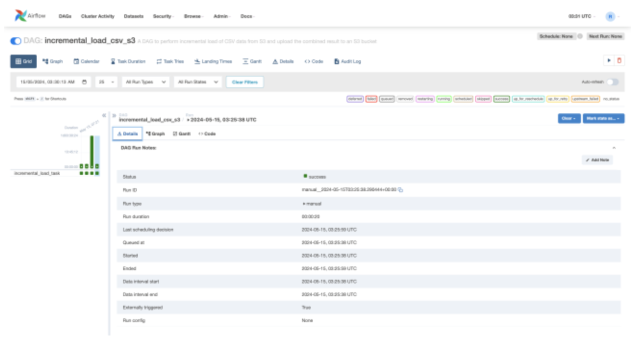
DAG: fetch_political_nytimes_data_S3 Fetch Political data from NY Times Archive and upload to S3

fetch_political_nytimes_data_S3

DAG Runs Summary	
Total Runs Displayed	6
Total success	6
First Run Start	2024-05-13, 03:54:07 UTC
Last Run Start	2024-05-14, 00:00:06 UTC
Max Run Duration	00:00:15
Mean Run Duration	00:00:11
Min Run Duration	00:00:08
DAG Summary	
Total Tasks	1
PythonOperator	1
DAG Details	
Dag id	fetch_political_nytimes_data_S3
Description	Fetch Political data from NY Times Archive and upload to S3
Fileloc	/usr/local/airflow/dags/DAG1Fetch.py
Has import errors	false
Has task concurrency limits	false
Is active	true
Is paused	true
Is subdag	false

Fig.7. Details and result of CSVDAG.py

2) Incremental Loading: CSVincrementalloadDAG.py (named as incremental_load.py in the Code Details Document): Following the data retrieval by `CSVDAG.py`,`CSVincrementalloadDAG.py` processes the CSV files located at `group-project-nyt/NYTData/year/month.csv`. This DAG performs incremental loads by comparing existing data in the staging area with the newly fetched data to update or append records without duplication. This ensures that only new or altered records are processed further, optimizing storage and processing resources. The output is staged appropriately for subsequent transformation tasks, maintaining data freshness and accuracy. Figure 8 shows the details and execution status of CSVincrementalloadDAG.py



DAG: incremental_load_csv_s3 A DAG to perform incremental load of CSV data from S3 and upload the combined result to an S3 bucket

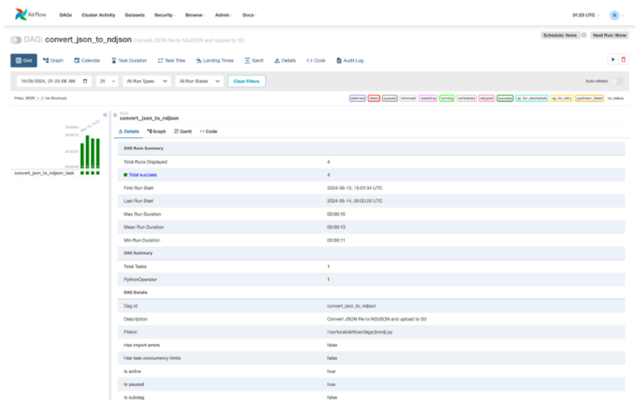
incremental_load_csv_s3 / 2024-05-15, 03:25:38 UTC

DAG Run Notes:

Status	success
Run ID	manual__2024-05-15T03:25:38.298444+00:00
Run type	manual
Run duration	00:00:20
Last scheduling decision	2024-05-15, 03:25:59 UTC
Queued at	2024-05-15, 03:25:38 UTC
Started	2024-05-15, 03:25:38 UTC
Ended	2024-05-15, 03:25:59 UTC
Data interval start	2024-05-15, 03:25:38 UTC
Data interval end	2024-05-15, 03:25:38 UTC
Externally triggered	True
Run config	None

Fig.8. Details and result of CSVincrementalloadDAG.py

3) JSON Transformation: jtondj.py (named as converting_json_to_ndjson.py in the Code Details Document): The `jtondj.py` DAG converts the updated CSV files from the path `group-project-nyt/NYTData/year/month.csv` into Newline Delimited JSON (NDJSON) format, a more suitable format for big data environments. This transformation is stored in `group-project-nyt/FinalBigQ.ndjson`, which simplifies the ingestion process for systems like Google BigQuery. This conversion is pivotal for enabling efficient large-scale data analysis and further processing in the `QueryReady.py` DAG. Figure 3 shows the details and execution status of converting_json_to_ndjson.py



DAG: convert_json_to_ndjson Convert JSON file to NDJSON and upload to S3

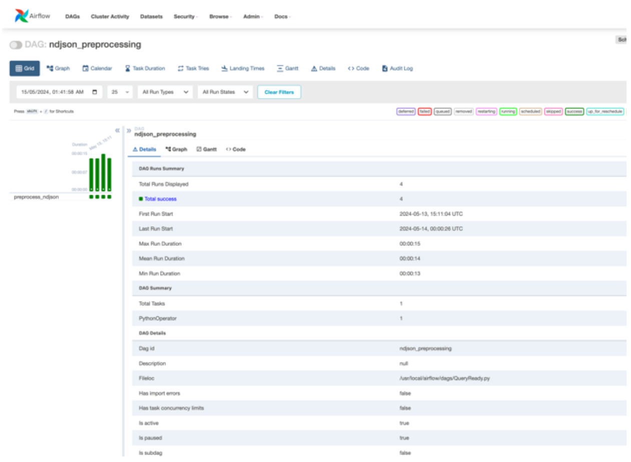
convert_json_to_ndjson

DAG Runs Summary	
Total Runs Displayed	4
Total success	4
First Run Start	2024-05-13, 15:01:34 UTC
Last Run Start	2024-05-14, 00:00:20 UTC
Max Run Duration	00:00:15
Mean Run Duration	00:00:13
Min Run Duration	00:00:11
DAG Summary	
Total Tasks	1
PythonOperator	1
DAG Details	
Dag id	convert_json_to_ndjson
Description	Convert JSON file to NDJSON and upload to S3
Fileloc	/usr/local/airflow/dags/jtondj.py
Has import errors	false
Has task concurrency limits	false
Is active	true
Is paused	true
Is subdag	false

Fig.9. Details and result of converting_json_to_ndjson.py

4) Query Preparation: QueryReady.py (named as pre-processing_ndjson.py in the Code

Details Document): `QueryReady.py` handles the final data preparations for analytical querying. It accesses the NDJSON file at `group-project-nyt/FinalBigQ.ndjson`, performing data cleansing, schema validation, and necessary transformations. These preprocessing steps ensure the data is optimized for performance and accuracy in querying environments such as Google BigQuery. The processed data is then stored back in `group-project-nyt`, ready for direct use in analytics and decision-making processes. Figure 4 shows the details and execution status of QueryReady.py



DAG
ndjson_preprocessing

DAG Runs Summary	
Total Runs Displayed	4
🟩 Total success	4
First Run Start	2024-05-13, 15:11:04 UTC
Last Run Start	2024-05-14, 00:00:26 UTC
Max Run Duration	00:00:15
Mean Run Duration	00:00:14
Min Run Duration	00:00:13
DAG Summary	
Total Tasks	1
PythonOperator	1
DAG Details	
Dag id	ndjson_preprocessing
Description	null
Fileloc	/usr/local/airflow/dags/QueryReady.py
Has import errors	false
Has task concurrency limits	false
Is active	true
Is paused	true
Is subdag	false

Fig.10. Details and result of QueryReady.py

5) Orchestration: MainDAG.py (named as schedule_dag_tasks.py in the Code Details Document): `MainDAG.py` serves as the central orchestrator for the entire data workflow, utilizing `TriggerDagRunOperator` to manage and sequence the execution of subsidiary DAGs. It triggers `CSVDAG.py`, `CSVincrementalloadDAG.py`, `jtondj.py`, and `QueryReady.py` based on a daily schedule. This DAG ensures that each subprocess is initiated only after the successful completion of the preceding process, thus maintaining the integrity and sequential flow of data operations. It effectively manages dependencies and timing within the AWS Managed Workflows for Apache Airflow (MWAA), ensuring that each data transformation step is correctly synchronized. Figure 5 shows the details and execution status of Main DAG.py

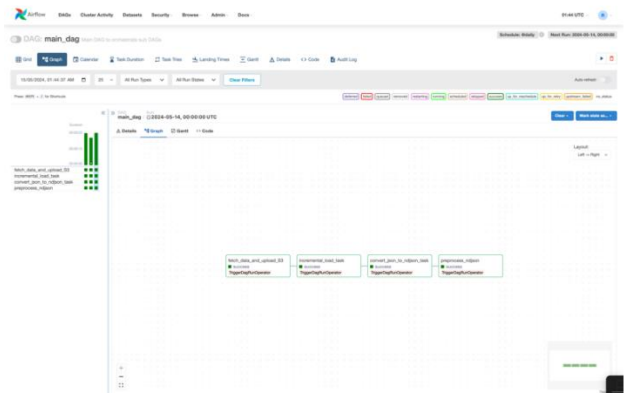


Fig.11. DAG task dependencies

C. *Datawarehouse Implementation*

1) Star Schema Model: The star schema implemented in our BigQuery data warehouse is a classical database design that optimizes query performance, simplifies data management, and enhances our data analytics capabilities. This schema is particularly suitable for handling complex queries, essential to our project's needs, allowing for efficient aggregation and fast data retrieval. Here's a detailed description of our schema components and their implementation.

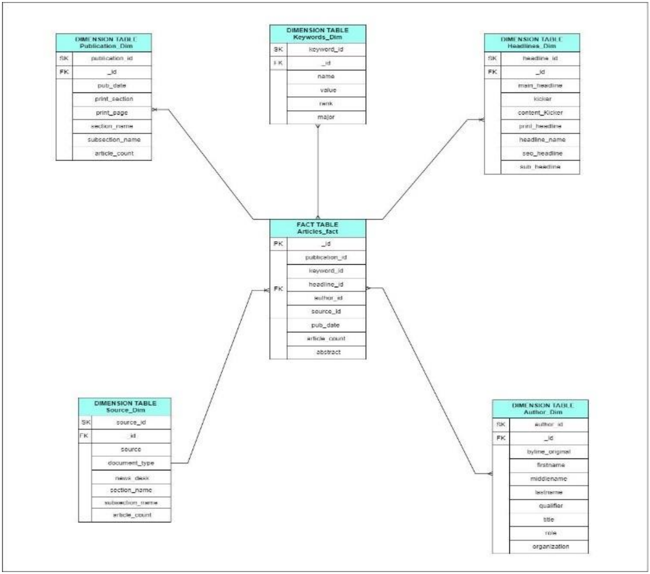


Fig.12. Implemented star schema model

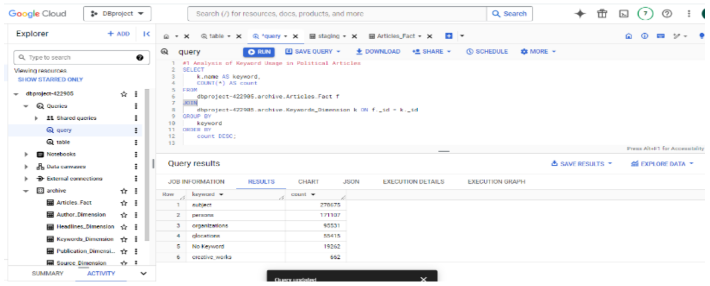
- Fact Table: Articles_fact is the central fact table in our star schema, designed to store quantitative metrics and key attributes about articles including a unique identifier (_id) as the primary key. This table links to various dimension tables via foreign keys (publication_id, keyword_id, headline_id, author_id, source_id), enabling detailed analysis across different attributes. It also includes the pub_date for time-series analysis, article_count for aggregative queries, and an abstract to provide a concise content summary, enhancing query performance and data accessibility in our BigQuery environment.

- Dimension Tables: The dimension tables in our star schema are Publication_Dim, Keywords_Dim, Headlines_Dim, Author_Dim, and Source_Dim which serve as detailed repositories for attributes related to articles, each enhancing the analytical depth of the data warehouse. These tables are structured to offer comprehensive metadata about articles, such as publication details, associated keywords, headline content, author information, and the publishing source. Each dimension table is linked to the Articles_fact table through foreign keys, enabling multifaceted queries that explore relationships and patterns within the data. For instance, the Author_Dim table contains fields like firstname, lastname, and role, allowing users to filter and aggregate data based on author-specific details. This structured approach not only enriches data analysis but also optimizes query performance by localizing attribute data within specific, well-defined tables.

2) Queries Implementation

<u>Query 1:</u> This query analyzes the frequency of keywords used in a database of political articles, counting each keyword's occurrences by joining the Articles_Fact and Keywords_Dimension tables

```
SELECT
    k.name AS keyword,
    COUNT(*) AS count
FROM
    dbproject-422905.archive.Articles_Fact
f
JOIN
    dbproject-
422905.archive.Keywords_Dimension   k   ON
f._id = k._id
GROUP BY
    keyword
ORDER BY
    count DESC;
```



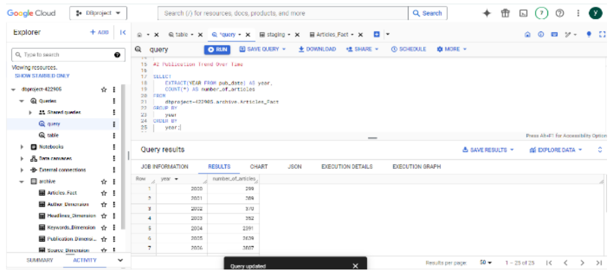
```
#1 Analysis of Keyword Usage in Political Articles
SELECT
    k.name AS keyword,
    COUNT(*) AS count
FROM
    dbproject-422905.archive.Articles_Fact f
JOIN
    dbproject-422905.archive.Keywords_Dimension k ON f._id = k._id
GROUP BY
    keyword
ORDER BY
    count DESC;
```

Row	keyword	count
1	subject	278675
2	persons	171107
3	organizations	95531
4	glocations	55415
5	No Keyword	19262
6	creative_works	662

Fig.13. Analysis of Keyword Usage in Political Articles

<u>QUERY 2:</u> This query calculates the number of articles published each year by extracting the year from the publication dates in the **Articles_Fact** table

```
SELECT
    EXTRACT(YEAR FROM pub_date) AS year,
    COUNT(*) AS number_of_articles
FROM
    dbproject-422905.archive.Articles_Fact
GROUP BY
    year
ORDER BY
    year;
```



```
#2 Publication Trend Over Time

SELECT
    EXTRACT(YEAR FROM pub_date) AS year,
    COUNT(*) AS number_of_articles
FROM
    dbproject-422905.archive.Articles_Fact
GROUP BY
    year
ORDER BY
    year;
```

Row	year	number_of_articles
1	2000	299
2	2001	309
3	2002	370
4	2003	352
5	2004	2391
6	2005	2639
7	2006	3607

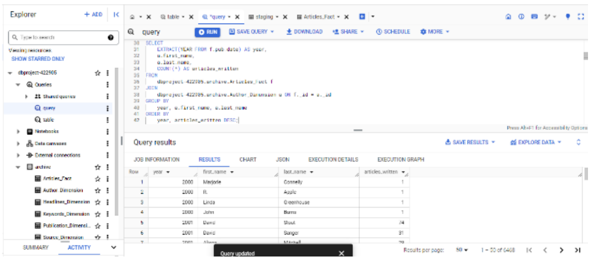
Results per page: 50 1 – 25 of 25

Fig.14. Publication trend over time

<u>QUERY 3:</u> This SQL query calculates the number of articles each author has written each year, by joining the **Articles_Fact** table with the **Author_Dimension** table. It groups the results by year, first name, and last name of the authors and orders them by year and the count of articles in descending order, showcasing the productivity of authors over the years.

```
SELECT
    EXTRACT(YEAR  FROM  f.pub_date)  AS
year,
    a.first_name,
    a.last_name,
    COUNT(*) AS articles_written
FROM
    dbproject-
422905.archive.Articles_Fact f
JOIN
```

```
        dbproject-
422905.archive.Author_Dimension  a  ON
f._id = a._id
GROUP BY
        year, a.first_name, a.last_name
ORDER BY
        year, articles_written DESC;
```



```
SELECT
    EXTRACT(YEAR FROM f.pub_date) AS year,
    a.first_name,
    a.last_name,
    COUNT(*) AS articles_written
FROM
    dbproject-422905.archive.Articles_Fact f
JOIN
    dbproject-422905.archive.Author_Dimension a ON f._id = a._id
GROUP BY
    year, a.first_name, a.last_name
ORDER BY
    year, articles_written DESC;
```

Row	year	first_name	last_name	articles_written
1	2000	Marjorie	Connelly	1
2	2000	R.	Apple	1
3	2000	Linda	Greenhouse	1
4	2000	John	Burns	1
5	2001	David	Stout	74
6	2001	David	Sanger	31

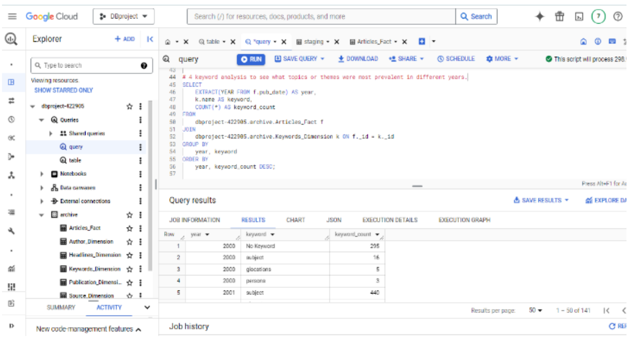
Query updated

Results per page: 50 1 – 50 of 6468

Fig.15. Analyzing Author Contributions of different authors have evolved over the years

QUERY 4: This SQL query extracts and counts the occurrences of keywords from articles by year, joining the **Articles_Fact** table with the **Keywords_Dimension** table. It groups the results by year and keyword, and then orders them by year and keyword count in descending order to highlight the most frequently used keywords each year.

```
SELECT
        EXTRACT(YEAR FROM f.pub_date)
AS year,
        k.name AS keyword,
        COUNT(*) AS keyword_count
FROM
        dbproject-
422905.archive.Articles_Fact f
JOIN
        dbproject-
422905.archive.Keywords_Dimension
k ON f._id = k._id
GROUP BY
        year, keyword
ORDER BY
        year, keyword_count DESC;
```



```
# 4 keyword analysis to see what topics or themes were most prevalent in different years.
SELECT
    EXTRACT(YEAR FROM f.pub_date) AS year,
    k.name AS keyword,
    COUNT(*) AS keyword_count
FROM
    dbproject-422905.archive.Articles_Fact f
JOIN
    dbproject-422905.archive.Keywords_Dimension k ON f._id = k._id
GROUP BY
    year, keyword
ORDER BY
    year, keyword_count DESC;
```

This script will process 298.

Row	year	keyword	keyword_count
1	2000	No Keyword	295
2	2000	subject	16
3	2000	glocations	5
4	2000	persons	5
5	2001	subject	440

Results per page: 50 1 – 50 of 141

Fig.16. Keyword analysis of topics which were most prevalent in different years

QUERY 5: This SQL query tracks monthly and yearly trends in election-related articles, summarizing total mentions and listing the top five election keywords by frequency for each period.

```
WITH KeywordRanks AS (
  SELECT
    EXTRACT(YEAR FROM pub_date)
AS year,
    EXTRACT(MONTH FROM pub_date)
AS month,
    k.value AS keyword,
    COUNT(*) AS keyword_count
  FROM
    dbproject-
422905.archive.Articles_Fact f
  JOIN
    dbproject-
422905.archive.Keywords_Dimension
k ON f._id = k._id
  WHERE
    k.value LIKE '%election%'
  GROUP BY
    year, month, keyword
)
SELECT
  year,
  month,
  SUM(keyword_count) AS
total_election_articles,
  ARRAY_AGG(keyword ORDER BY
keyword_count DESC LIMIT 5) AS
top_keywords
FROM
  KeywordRanks
GROUP BY
  year, month
ORDER BY
  year, month;
```

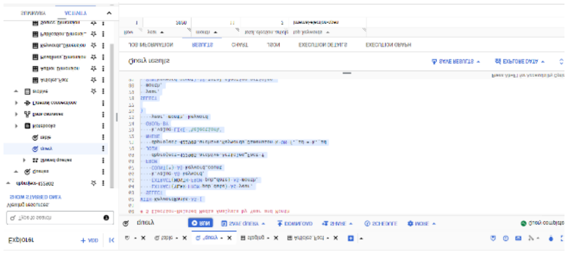
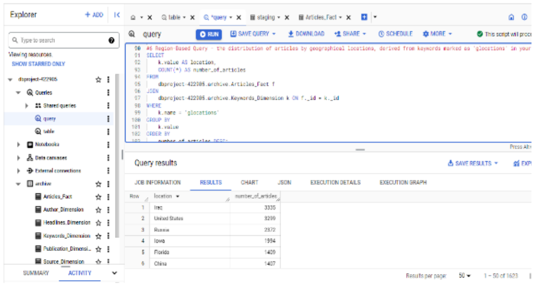


Fig.17. Election-Related Media Analysis by Year and Month

QUERY 6: This SQL query calculates the monthly count of articles related to "election" keywords by year and month, using a Common Table Expression (CTE) to first filter and group the data. The main query then aggregates these counts and lists the top five most frequent election-related keywords for each month, ordered chronologically.

```
SELECT
    k.value AS location,
    COUNT(*) AS number_of_articles
FROM
    dbproject-
422905.archive.Articles_Fact f
JOIN
    dbproject-
422905.archive.Keywords_Dimension    k
ON f._id = k._id
WHERE
    k.name = 'glocations'
GROUP BY
    k.value
ORDER BY
    number_of_articles DESC;
```



```
#6 Region-Based Query - the distribution of articles by geographical locations, derived from keywords marked as 'glocations' in your
SELECT
    k.value AS location,
    COUNT(*) AS number_of_articles
FROM
    dbproject-422905.archive.Articles_Fact f
JOIN
    dbproject-422905.archive.Keywords_Dimension k ON f._id = k._id
WHERE
    k.name = 'glocations'
GROUP BY
    k.value
ORDER BY
    number_of_articles DESC;
```

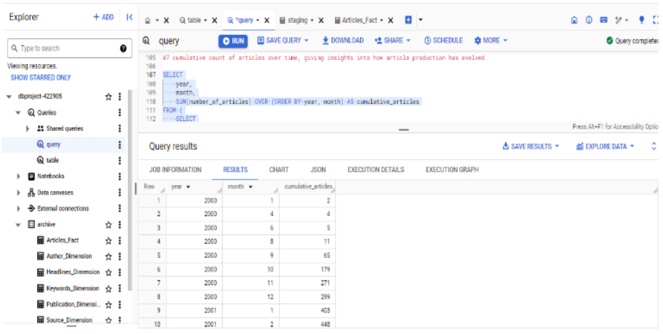
Row	location	number_of_articles
1	Iraq	3335
2	United States	3299
3	Russia	2372
4	Iowa	1994
5	Florida	1409
6	China	1407

Results per page: 50 1 – 50 of 1623

Fig.18.The distribution of articles by geographical locations

QUERY 7: This query calculates the cumulative total of articles published each month, grouping them by year and month. It displays a running total ordered chronologically to show the growth in article publication over time.

```
SELECT
    year,
    month,
    SUM(number_of_articles) OVER
(ORDER BY year, month) AS
cumulative_articles
FROM (
    SELECT
        EXTRACT(YEAR FROM pub_date)
AS year,
        EXTRACT(MONTH FROM pub_date)
AS month,
        COUNT(*) AS
number_of_articles
    FROM
        dbproject-
422905.archive.Articles_Fact
    GROUP BY
        year, month
)
ORDER BY
    year, month;
```



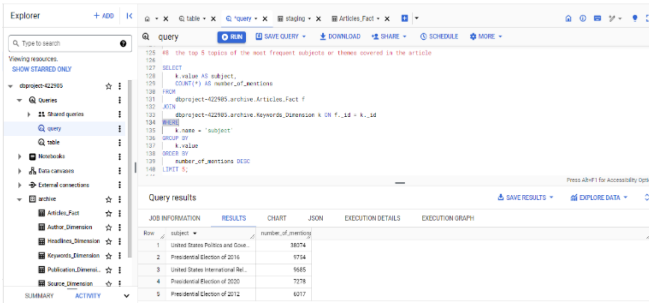
```
#7 cumulative count of articles over time, giving insights into how article production has evolved

SELECT
    year,
    month,
    SUM(number_of_articles) OVER (ORDER BY year, month) AS cumulative_articles
FROM (
    SELECT
```

Row	year	month	cumulative_articles
1	2000	1	2
2	2000	4	4
3	2000	6	5
4	2000	8	11
5	2000	9	65
6	2000	10	179
7	2000	11	271
8	2000	12	299
9	2001	1	403
10	2001	2	448

Fig.19.Cumulative count of articles over time

QUERY 8: This query retrieves the top five most frequently mentioned subjects in articles, counting mentions and ordering them in descending frequency.

```
SELECT
    k.value AS subject,
    COUNT(*) AS number_of_mentions
FROM
    dbproject-
422905.archive.Articles_Fact f
JOIN
    dbproject-
422905.archive.Keywords_Dimension    k
ON f._id = k._id
WHERE
    k.name = 'subject'
GROUP BY
    k.value
ORDER BY
    number_of_mentions DESC

LIMIT 5;
```



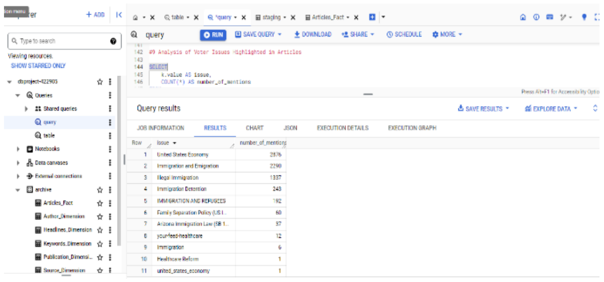
```
#8  the top 5 topics of the most frequent subjects or themes covered in the article

SELECT
    k.value AS subject,
    COUNT(*) AS number_of_mentions
FROM
    dbproject-422905.archive.Articles_Fact f
JOIN
    dbproject-422905.archive.Keywords_Dimension k ON f._id = k._id
WHERE
    k.name = 'subject'
GROUP BY
    k.value
ORDER BY
    number_of_mentions DESC
LIMIT 5;
```

Row	subject	number_of_mentions
1	United States Politics and Gove...	38074
2	Presidential Election of 2016	9754
3	United States International Rel...	9685
4	Presidential Election of 2020	7278
5	Presidential Election of 2012	6017

Fig.20. Top 5 topics of the most frequent subjects or themes covered in the article

QUERY 9: This query counts mentions of specific issues (healthcare, immigration, economy) in articles, grouping results by issue and ordering them by frequency of mentions in descending order.

```
SELECT
    k.value AS issue,
    COUNT(*) AS number_of_mentions
FROM
    dbproject-
422905.archive.Articles_Fact f
JOIN
    dbproject-
422905.archive.Keywords_Dimension   k
ON f._id = k._id
WHERE
    k.name = 'subject' AND
    (LOWER(k.value)              LIKE
'%healthcare%' OR
    LOWER(k.value)               LIKE
'%immigration%' OR
    LOWER(k.value) LIKE '%economy%')
GROUP BY
    k.value
ORDER BY
    number_of_mentions DESC;
```



```
#9 Analysis of Voter Issues Highlighted in Articles

SELECT
    k.value AS issue,
    COUNT(*) AS number_of_mentions
```

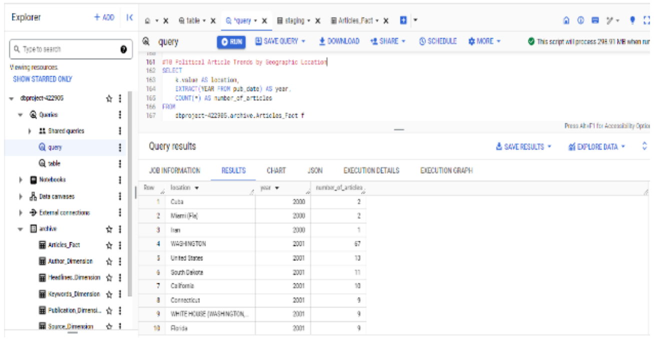
Query results

Row	issue	number_of_mentions
1	United States Economy	2376
2	Immigration and Emigration	2290
3	Illegal Immigration	1337
4	Immigration Detention	243
5	IMMIGRATION AND REFUGEES	192
6	Family Separation Policy (US I...	60
7	Arizona Immigration Law (SB 1...	37
8	your-feed-healthcare	12
9	Immigration	6
10	Healthcare Reform	1
11	united_states_economy	1

Fig.21. Analysis of Voter Issues Highlighted in Articles

<u>QUERY 10:</u> This SQL query counts articles mentioning specific geographic locations each year, grouping by location and year, and sorting by the number of mentions in descending order.

```
SELECT
    k.value AS location,
    EXTRACT(YEAR FROM pub_date) AS
year,
    COUNT(*) AS number_of_articles
FROM
        dbproject-
    422905.archive.Articles_Fact f
JOIN
    dbproject-
422905.archive.Keywords_Dimension k
ON f._id = k._id
WHERE
    k.name = 'glocations'
GROUP BY
    location, year
ORDER BY
    year, number_of_articles DESC;
```



```
#10 Political Article Trends by Geographic Location
SELECT
    k.value AS location,
    EXTRACT(YEAR FROM pub_date) AS year,
    COUNT(*) AS number_of_articles
FROM
    dbproject-422905.archive.Articles_Fact f
```

Query results

Row	location	year	number_of_articles
1	Cuba	2000	2
2	Miami (Fla)	2000	2
3	Iran	2000	1
4	WASHINGTON	2001	67
5	United States	2001	13
6	South Dakota	2001	11
7	California	2001	10
8	Connecticut	2001	9
9	WHITE HOUSE (WASHINGTON,...	2001	9
10	Florida	2001	9

Fig.22. Political Article Trends by Geographic Location

VI. BUSINESS INTELLIGENCE

To visualize data from BigQuery in Power BI, we created a connection between the two platforms. This integration involved setting up the Power BI service to access our BigQuery dataset using a native connector, which allows for seamless querying and data retrieval. After configuring the necessary permissions, we imported the desired datasets into Power BI. This setup allowed us to utilize Power BI's data visualization tools to create dynamic reports and dashboards based on the combined data stored in BigQuery.

A. Analyzing U.S. Political Discourse: A Word Cloud Visualization

The visualization is a word cloud that highlights key terms such as "Trump," "Obama," "Senate," and "Republican," indicating an analysis of topics pertinent to American politics. It serves as a tool for visualizing the frequency and relevance of these terms in discussions, potentially useful for identifying major themes in political texts.

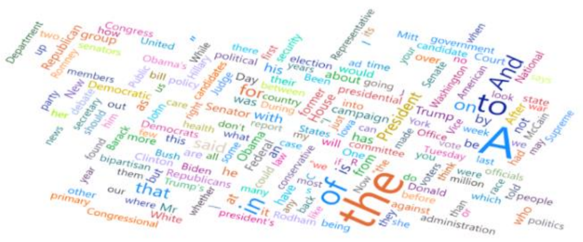


Fig.23. A word Cloud Visualization

B. Global Operations Map: Strategic Insights and Resource Distribution

The visualization serves as a strategic tool by showcasing the global presence and operational reach of The New York Times through a visual map, indicating physical or reporting locations. It aids in resource allocation and strategic planning by helping to manage logistical operations and staff deployment effectively

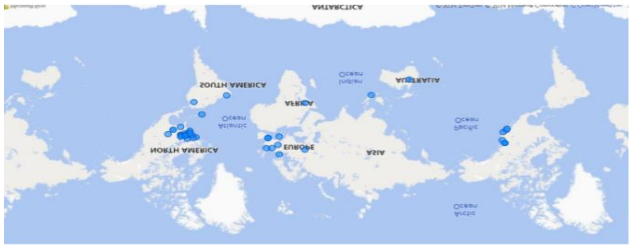


Fig.24. *Global Operations Map*

C. Content Volume and Specialization Across News Desks

The visualization differentiates between high-volume content producers like the "WEB" desk and specialized areas like "Voter Guide 2004" and "Travel Desk," highlighting both dominant and niche reporting areas. It also emphasizes a strong editorial focus on U.S. politics, as evidenced by multiple "Washington" labeled desks.

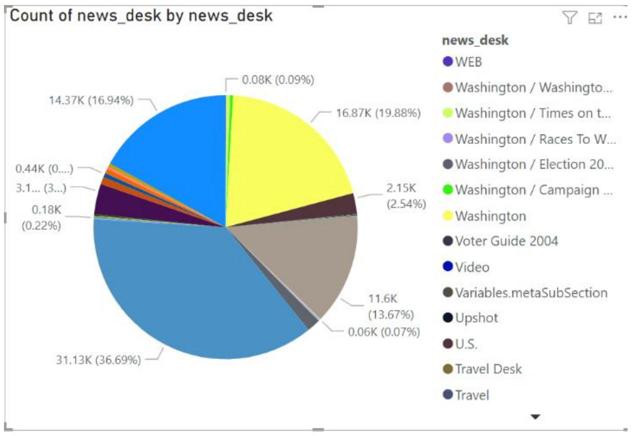


Fig 25. Content Volume and Specialization Across News Desks

D. Dashboard of Linegraph and Piechart

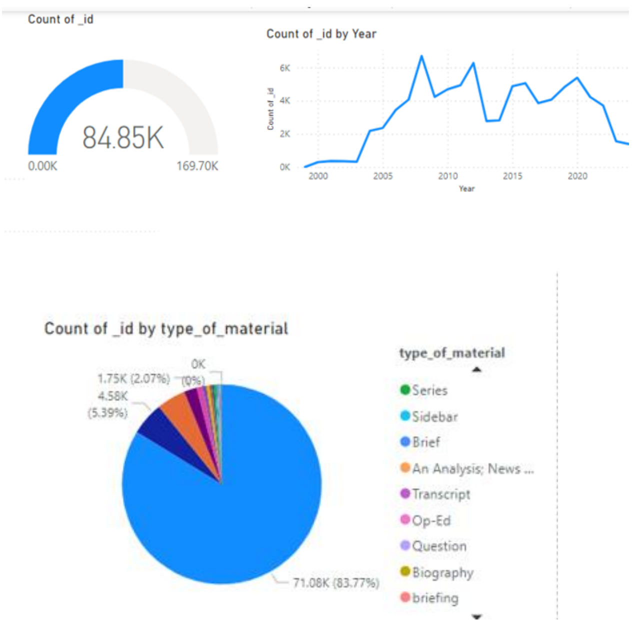


Fig 26. Overview of Trends - Dashboard

The line chart depicts the trend of article counts (_id) over the years, indicating fluctuations in publishing activity across different time periods.The line chart in the first image shows that the number of articles published peaked around 2008-2009 and then witnessed a decline in subsequent years. This pattern could potentially reflect changes in the news cycle or other factors influencing publishing activity during those time periods.

The pie chart is used to visualize the distribution of articles across different types of material (type_of_material) in the dataset. The majority of articles (83.77%) are classified as "News," followed by other categories. It highlights the dominance of "News" articles in the data.

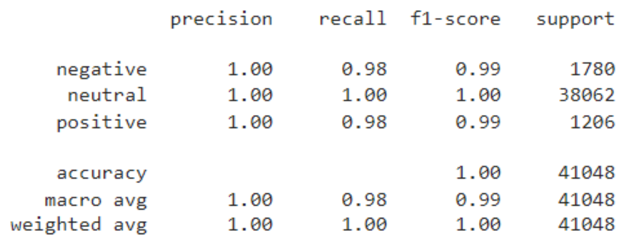
VII. Statistical Analysis

A. Model I - Sentiment Analysis Based on the 'keywords' column:

The model implementation begins with a data cleaning process that involves techniques such as tokenization, removal of stopwords, lowercasing, stemming, lemmatization, punctuation elimination, and extraction of numerical data, URLs, and HTML tags. This ensures that the text data is standardized and free from irrelevant noise, helps in setting up a proper foundation for subsequent analysis. Further, sentiment labeling is performed by generating polarity scores using TextBlob's sentiment analysis model, categorizing the texts as positive, negative, or neutral based on their sentiment scores. This sentiment-labeled dataset is then used to train a Logistic Regression Model for sentiment classification. The model first undergoes a process data loading, followed by feature extraction via TF-IDF vectorization, data splitting for training and testing, model training, prediction of sentiment labels for the test data, and finally model evaluation using precision, recall, and F1-score metrics. Additionally, correlation analysis and hypothesis testing are conducted to explore the relationship between word count and sentiment scores, providing insights into potential patterns in textual expression and sentiment perception.

This sentiment analysis model will help us understand public sentiment towards political figures, policies, and events, providing valuable insights for election prediction system. By accurately classifying texts as positive, negative, or neutral, the model enables analysts to record public opinion and preferences, guiding them through strategic decision-making for political campaigns. In order to understand the relationship between variables like word count and sentiment scores, regression analysis is used. By determining whether observed variations in variables are statistically significant, hypothesis helps us understand patterns of sentiment expression.

1) Results and Interpretation:



```
              precision    recall  f1-score   support

    negative       1.00      0.98      0.99      1780
     neutral       1.00      1.00      1.00     38062
    positive       1.00      0.98      0.99      1206

    accuracy                           1.00     41048
   macro avg       1.00      0.98      0.99     41048
weighted avg       1.00      1.00      1.00     41048
```

Fig.27. Result of the Logistic Regression Model

Precision:

- Precision, Recall, F1-Score:

Negative Class: Precision of 1.00 indicates perfect accuracy in identifying negative instances.

Neutral Class: Precision of 1.00 signifies all predicted neutral instances were correct.

Positive Class: Precision of 1.00 indicates accurate identification of positive instances.

Negative Class: Recall of 0.98 suggests 98% of actual negative instances were correctly identified.

Neutral Class: Recall of 1.00 indicates all actual neutral instances were correctly identified.

Positive Class: Recall of 0.98 suggests 98% of actual positive instances were correctly identified.

F1 Score:

Negative Class: F1 score of 0.99 demonstrates high performance in identifying negative instances.

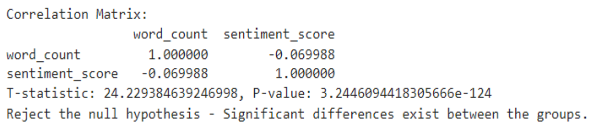
Neutral Class: F1 score of 1.00 reflects perfect performance in identifying neutral instances.

Positive Class: F1 score of 0.99 indicates balanced and high performance in identifying positive instances.

Correlation Analysis and Hypothesis Testing:

Null Hypothesis (H0): There is no difference in the mean word counts between positive and neutral sentiments.

Alternative Hypothesis (H1): There is a significant difference in the mean word counts between positive and neutral sentiments.



```
Correlation Matrix:
               word_count  sentiment_score
word_count       1.000000        -0.069988
sentiment_score -0.069988         1.000000
T-statistic: 24.229384639246998, P-value: 3.2446094418305666e-124
Reject the null hypothesis - Significant differences exist between the groups.
```

Fig.28. Correlation Matrix

- **Correlation Matrix Analysis**

The value indicates a slight negative correlation between the number of words in a text and the sentiment score assigned to it. The negative value suggests that as texts become longer, they might slightly tend to be less positive. However, the magnitude of this correlation is very small, close to zero, indicating that the relationship between word count and sentiment is very weak.

While there is a statistically observable relationship, it is not strong enough to suggest that word count significantly influences sentiment, or that sentiment can be accurately predicted based on word count alone.

- **T-Test Results**

T-statistic: 24.229384639246998

P-value: 3.24460944183056.

The T-statistic is quite high, suggesting a significant distance between the means of the two groups.

The P-value is exceedingly small (practically zero), far below the typical threshold for significance (e.g., 0.05 or 0.01). This confirms that the observed differences are extremely unlikely to have occurred by chance.

There is a statistically significant difference in word counts between positive and neutral sentiment texts, meaning that one group consistently has higher or lower word counts than the other, which could indicate stylistic or substantive differences in how different sentiments are expressed. Hence the Null Hypothesis was Rejected**.**

B. *Model II - Sentiment Analysis based on the ‘abstract’ column*

The second model implements sentiment analysis based on the ‘abstract’ column. To develop this model, we have first used Latent Dirichlet Allocation (LDA) to identify recurring themes in news stories mentioning the presidential contenders by extracting popular subjects from a corpus of text data. The method uses CountVectorizer to transform text data into a term-document matrix, which aids in locating frequently occurring terms important in the data's context. The extracted themes include a wide range of political arenas, including elections, policy, and international relations, as well as the personal characteristics and behaviors of the candidates. For Instance, in Figure 38, which represents the result of Topic Modeling indicates that Topic #5 heavily features Joe Biden in multiple contexts, implying considerable coverage or conversations surrounding his political movements or utterances. In contrast, Topic #0 centers around legal and judicial issues, indicating talks on legal procedures or legislation.

Based on these results, we identified that performing Sentiment Analysis on the Presential Candidates names, Trump and Biden, would allow us to extract sentiment data and effectively predict electoral results for the upcoming Presidential Elections in 2024.

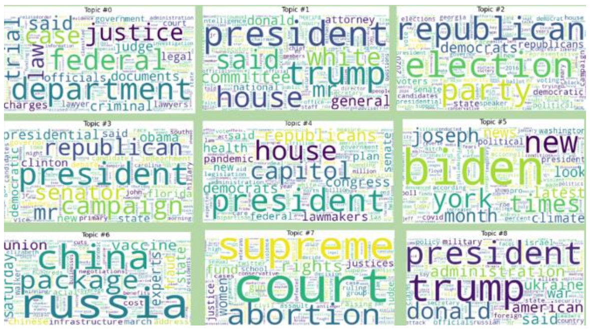


Fig.29. Result of Topic Modeling

The sentiment analysis was performed to gauge the public and media sentiment related to articles mentioning Biden and Trump (referred from the word cloud) over several years, using the VaderSentiment analyzer. Correlating public opinion with electoral outcomes and candidate popularity is the goal of this analysis. Each article's sentiment score is determined, and the average daily sentiment is plotted over time to track changes and trends in the public perception. Figure 39 represents the result of the Sentiment Analysis based on the ‘abstract’ column with notable swings seen around the 2016 and 2020 elections, the graphic effectively illustrates how sentiment fluctuates before, during, and after election years.

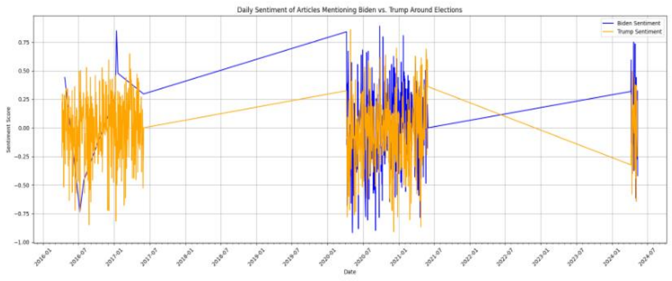


Fig.30. Result of Sentiment Analysis based on the ‘abstract’ column

1) Results and Interpretations:

- Historical Trends:
An examination of previous election years indicates a relationship between the election results and the sentiment trends. For instance, a move toward Biden in 2020 related to his electoral triumph, while a higher degree of positivity toward Trump in 2016 was correlated with his election victory.
- Current Year Analysis:
The analysis states that there is not enough information available for this year to make a firm prediction about the results of future elections.

This highlights the need for continuous data monitoring to better understand the evolving public opinion as the election date approaches.

- Strategic Insights: Based on the results, it is advised to increase sentiment research and media monitoring as the election approaches, concentrating on important topics identified in the topic modeling to assess voter sentiment and campaign efficacy more accurately.

VIII. CONCLUSION

The PolitiCast Election Prediction System successfully orchestrated a comprehensive data pipeline that integrated the vast New York Times dataset with real-time data. The combination of Apache Airflow for workflow management, MongoDB for efficient data storage, and Google BigQuery for high-performance data warehousing gave us a solid structure. Airflow's scheduling and monitoring capabilities ensured that each stage of the ETL process, from data extraction to transformation and loading, is executed flawlessly and in a timely manner.

The project's data pipeline demonstrated adaptability by handling both archived and real-time data. The integration of the New York Times Archive API facilitated the continuous ingestion of up-to-date information, while the incremental loading process optimized resource utilization by updating only new or modified records. The star schema model implemented within BigQuery, coupled with the efficient querying capabilities of SQL, enabled multifaceted analysis and helped us identify intricate patterns within the data. Power BI's integration further enhanced the project's analytical aspects, facilitating the creation of dashboards and reports that conveyed insights through visualizations. The project's statistical analysis component, involving sentiment analysis models and correlation studies, displayed invaluable insights into public dynamics and their potential influence on election outcomes.