TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

TECHNICAL REPORT

DATA PIPELINES ENGINEERING USING APACHE AIRFLOW AND AMAZON WEB SERVICES

Reddit End-to-end Data Engineering Project

Author 1:
Moetez Abbassi

Author 2:
Saif Chouaya

February, 2025

ABSTRACT

The ”Reddit End-to-End Data Engineering Project” is a robust, scalable solution designed to automate the extraction, transformation, and analysis of Reddit data using a structured data pipeline. By integrating modern data engineering tools and cloud services, this pipeline ensures efficient data processing, orchestration, and storage for analytics and business intelligence.

This project follows the ETL (Extract, Transform, Load) workflow, leveraging a combination of open-source frameworks and AWS cloud services to streamline data ingestion, transformation, and querying.

Contents

ACRONYMS

- **ETL** - Extract, Transform, Load
- **API** - Application Programming Interface
- **AWS** - Amazon Web Services
- **S3** - Simple Storage Service
- **SQL** - Structured Query Language
- **DAG** - Directed Acyclic Graph
- **UI** - User Interface

List of Figures

CHAPTER 1: Introduction

1.1 Overview

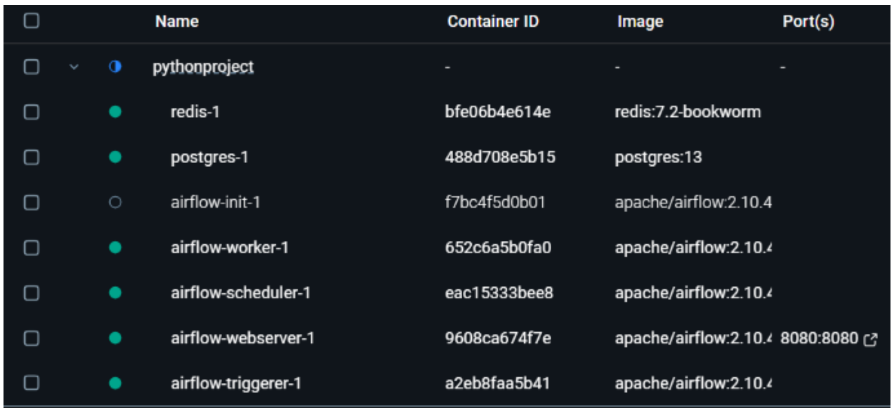
This report presents the creation of a seamless ETL process by the integration of several tools and software including Docker, Apache Airflow, Celery, PostgreSQL, S3 Buckets, AWS Glue, Athena, and Redshift. The main goal of this project is to automate the data collection and processing workflow in order to store and analyze real-time Reddit data efficiently. GitHub repositories for this project:
https://github.com/MoetezAbbassi/RedditPipeLineProject
https://github.com/ChouayaSaif/RedditDataPipelineProject

1.2 Project Setup

To orchestrate the data pipelines, W have set up a workflow using Apache Airflow with Celery Executor. The overall workflow was initiated inside a Docker virtual environment.
The process starts with configuring a ”docker-compose.yml” file, which is mainly downloaded from the official Apache Airflow website and edited to match the goal of my project. The content of ”docker-compose.yml” includes the characteristics of the Airflow environment, which contains the following items:
- **x-airflow-common:** Ensures consistency of multiple Airflow Services by defining common configurations.
- **Postgres Service**: The back-end database for Airflow, which will store the data and record the executions.
- **Redis Service**: Redis acts as a message broker for Celery, facilitating communication between the Airflow scheduler and Celery workers. When the Airflow scheduler assigns a task, it sends a message to Redis. Celery workers then pick up the task from Redis and execute it.
- **Airflow Webserver**: The UI to manage Airflow DAGs and tasks.
- **Airflow Scheduler:** Continuously scan the DAG file to ensure that Airflow runs in accordance with the required configurations.
- **Airflow Worker:**: Execute tasks given by Celery in a scalable manner.
- **Airflow Trigger:** Orders the scheduling and execution of tasks by conditions or events that determine when a task should be executed within a DAG.

- **Volumes:** Stores the Postgres database and allows communication between containers.
- **Airflow Init:** Initializes the Database, creates the root user followed by the needed permissions.

Then, the images of Redis, Postgres, and Apache Airflow are built using Docker, along with the needed requirements from ”requirements.txt” by using the command ”docker compose up -d –build”. The Airflow server will be hosted and we will get the following interface:



Name	Container ID	Image	Port(s)
pythonproject	-	-	-
redis-1	bfe06b4e614e	redis:7.2-bookworm	
postgres-1	488d708e5b15	postgres:13	
airflow-init-1	f7bc4f5d0b01	apache/airflow:2.10.4	
airflow-worker-1	652c6a5b0fa0	apache/airflow:2.10.4	
airflow-scheduler-1	eac15333bee8	apache/airflow:2.10.4	
airflow-webserver-1	9608ca674f7e	apache/airflow:2.10.4	8080:8080
airflow-triggerer-1	a2eb8faa5b41	apache/airflow:2.10.4	

Figure 1: Docker UI: Building Images and running the Webserver with Docker

CHAPTER 2 System Design and Architecture

2.1 Functional Requirements

- **Data Ingestion:**
  - The system must connect to the Reddit API to fetch posts and comments.
  - It must support scheduled and real-time data extraction.
  - Extracted data should be temporarily stored in PostgreSQL before processing.
- **Data Transformation:**
  - The system must move raw data to Amazon S3 for storage.
  - AWS Glue must process and catalog the data, making it queryable.
  - The data must be transformed using Amazon Athena for structured analysis.
- **Data Storage:**
  - Transformed data should be loaded into Amazon Redshift for analytics.
  - Metadata and workflow states must be tracked in PostgreSQL.
- **Task Orchestration:**
  - Apache Airflow must orchestrate the ETL process using Directed Acyclic Graphs (DAGs).
  - Celery should enable asynchronous task execution for scalability.
  - Docker must be used to containerize the system for easy deployment.
- **Data Querying Analysis:**
  - Amazon Athena must allow users to query raw and processed data.
  - Data in Redshift should be available for further analysis and business intelligence.
- **System Monitoring and Logging:**

  - The system must log all processes, including data ingestion, transformation, and loading.
  - Airflow's UI must allow users to monitor workflow execution.

2.2 Non-Functional Requirements

- **Scalability**
  - The system should handle large-scale Reddit data without performance degradation.
  - AWS services should be leveraged for auto-scaling as data volume increases.
- **Performance**
  - Data extraction should be optimized to prevent exceeding Reddit API rate limits.
  - The ETL process should complete within a defined SLA (e.g., daily processing within 1 hour).
- **Security**
  - API credentials must be securely stored (e.g., environment variables, AWS Secrets Manager).
  - Access to AWS services should be restricted based on roles and permissions.
- **Maintainability**
  - The system should allow easy modifications for new data sources.
  - It must support modular architecture to add new transformations or storage solutions.
- **Portability**
  - The system should be deployable on any cloud environment with minimal changes.
  - Docker should enable cross-platform compatibility.

2.3 System Architecture

The architecture of the Reddit Data Engineering Project is structured to handle large volumes of data systematically. The process initiates with data extraction from Reddit, followed by transformation and loading into the data warehouse. Each component plays a pivotal role in maintaining the efficiency and reliability of the pipeline.

Key Components and Their Roles

Data Extraction Utilizing Reddit's API, data is extracted and stored in a PostgreSQL database. This step ensures that raw data is collected and made available for subsequent processing.

Task Orchestration with Apache Airflow Airflow manages and schedules the various tasks within the pipeline. It ensures that data extraction, transformation, and loading processes are executed in the correct sequence and at the appropriate times.

Asynchronous Task Execution with Celery Celery works in conjunction with Airflow to handle asynchronous tasks, enhancing the pipeline's ability to manage multiple operations concurrently and improving overall efficiency.

Data Storage in Amazon S3 After extraction, data is stored in Amazon S3, providing a scalable and durable storage solution. This setup allows for easy access and retrieval of data during the transformation phase.

Data Transformation with AWS Glue AWS Glue processes the raw data stored in S3, transforming it into a structured format suitable for analysis through a well-conceived visual ETL process. This service automates extract, transform, and load (ETL) operations, reducing the need for manual intervention.

Data Analysis with Amazon Athena Athena enables users to perform SQL queries on the transformed data directly in S3. This serverless query service allows for quick and cost-effective data analysis without the need to load data into a traditional database.

Data Warehousing in Amazon Redshift The final, transformed data is loaded into Amazon Redshift, a fully managed data warehouse service. Redshift facilitates complex queries and analysis, supporting data-driven decision-making processes.

Data Flow Process

The data flow within the Reddit Data Engineering Project follows a structured path:

1. **Extraction**: Data is extracted from Reddit's API and stored in PostgreSQL.
2. **Storage**: The extracted data is then moved to Amazon S3 for scalable storage.
3. **Transformation**: AWS Glue processes and transforms the data into a structured format.
4. **Loading**: The transformed data is loaded into Amazon Redshift for advanced analysis.
5. **Analysis**: Users can perform queries on the data using Amazon Athena or directly within Redshift, enabling comprehensive data analysis and reporting.

This architecture ensures a robust and efficient pipeline for managing Reddit data, from extraction to analysis, leveraging the strengths of each integrated service.

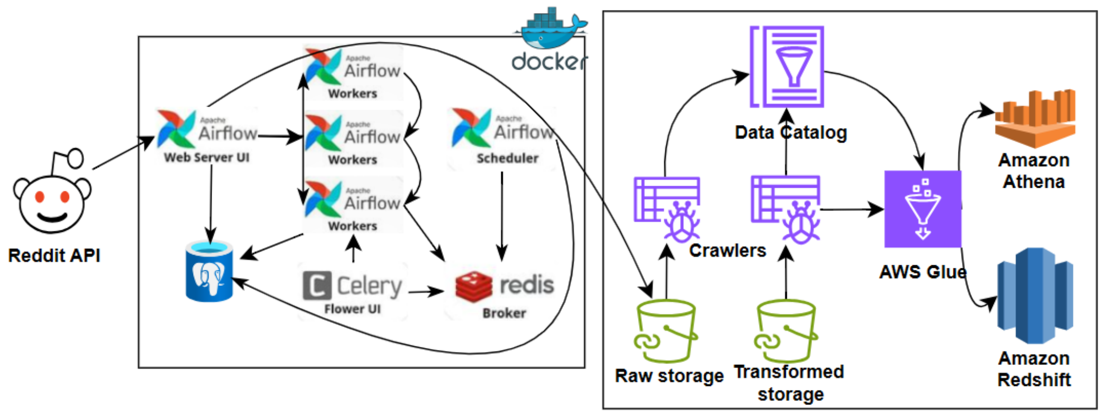


Figure 2: System Architecture

Apache Airflow Workflow

Apache Airflow provides a framework for defining workflows as Directed Acyclic Graphs (DAGs) and executing tasks efficiently.

Step 1: DAG Execution Initiation

The Scheduler detects the DAG file `reddit_dag.py` for execution, reads it, and determines the tasks to be executed. This file performs the following:

- **Extract data from Reddit**: Uses `reddit_pipeline` to fetch posts from the `dataengineering` subreddit.
- **Store extracted data**: Saves the extracted data with a timestamped filename.
- **Upload to AWS S3**: Uses `upload_s3_pipeline` to transfer the extracted data to an S3 bucket.

This ensures a fully automated data pipeline that runs daily (`@daily`), extracting data and pushing it to cloud storage.

Step 2: Task Queueing

Each task in the DAG file is pushed to the Redis queue. The Celery broker acts as a middleman and holds tasks in the queue until a worker picks them up.

Step 3: Task Execution

- Celery workers run in the Airflow Worker container.
- A worker continuously polls Redis for new tasks.
- Once a task (e.g., `extract_data`) is available, the worker picks it up and starts execution.

Step 4: Running the Task

- The worker runs the Python function associated with the task.
- The `extract_data` task calls the `reddit_pipeline` function from `pipelines.reddit` which:
  - Calls Reddit's API.
  - Retrieves data.
  - Saves it as a CSV file (likely in `/opt/airflow/data/`).

Step 5: Task Completion and Status Update

- Once the task is completed, the worker sends the task status (success or failure) to the Metadata DB (PostgreSQL).
- The Scheduler reads this information and decides what to do next.

Step 6: Dependency Management

- If the DAG has multiple tasks, the Scheduler checks dependencies.
- If `extract_data` is successful, the next task (e.g., `load_data` or `process_data`) is queued in Redis.
- The cycle repeats:
  - Redis queues the task.
  - Celery worker picks it up.
  - The task is executed.
  - The status is updated in PostgreSQL.

Step 7: DAG Completion

- Once all tasks in the DAG have finished successfully:
  - The final status of the DAG is updated in the Metadata DB.
  - The result can be viewed in the Airflow Web UI.

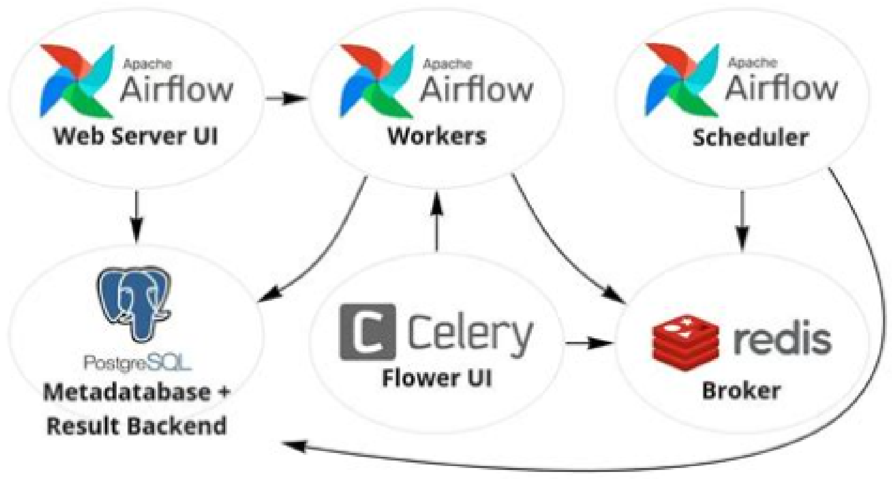


Figure 3: Apache Airflow Workflow

CHAPTER 3 Reddit ETL Process

The first ETL process is included in the ”reddit-etl.py” file, and it has these following items:

- Extraction:

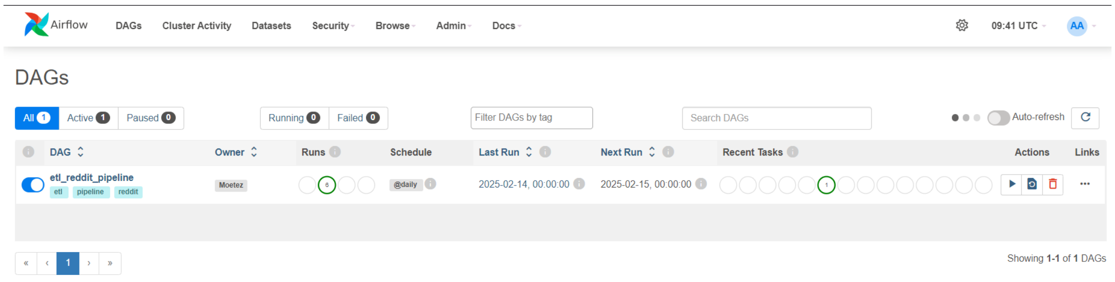
First, we login into Reddit and create an app with the type ’script’ to retrieve the data. We will receive secret keys to access Reddit’s API from our app, and then we will add these keys to our configuration file to ensure the automation of the extraction process. We use the ”praw” library in Python to fetch the reddit data from a subreddit of our choice (dataengineering subreddit in this project) The obtained dataframe will include several features such as comments, posts, date, etc.

- Transformation:

The reddit data will be simply redesigned to ensure consistency within the data and make it easier to manipulate in the AWS.

- Loading:

We obtain the final dataframe which will be automatically processed by Airflow, and it will be available almost instantly in the Webserver:



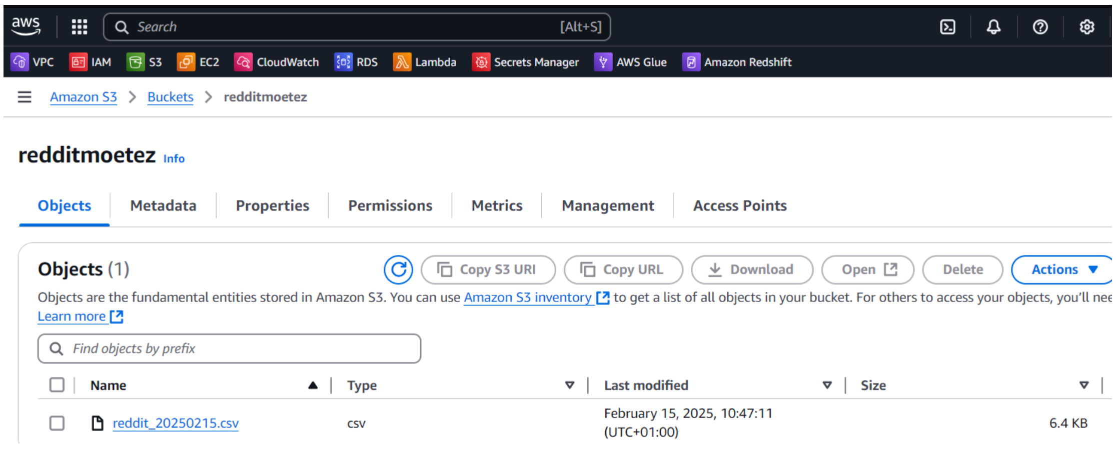
DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
etl_reddit_pipeline etl pipeline reddit	Moetez	6	@daily	2025-02-14, 00:00:00	2025-02-15, 00:00:00	1		···

Figure 4: Airflow Webserver after executing DAG file

CHAPTER 4 Data Processing With AWS:

During this phase, we will be making a pipeline to connect our Airflow DAG to an S3 Bucket in Amazon Web Services.

We start by getting our AWS confidential keys and adding them to the configuration file. Then, the ”aws-etl.py” automatically connects an S3 bucket to our Airflow DAGs, fetches our final dataframe and stores it as a CSV file.



Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll nee

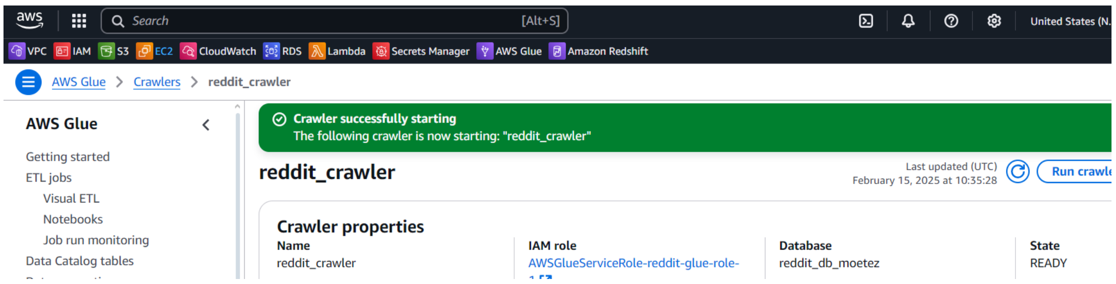
Name	Type	Last modified	Size
reddit_20250215.csv	csv	February 15, 2025, 10:47:11 (UTC+01:00)	6.4 KB

Figure 5: Uploading Data to S3 Bucket

AWS Glue can create a Crawler that catalogs the data stored in S3 bucket, it also has another internal ETL Process to prepare the data for further analysis.

The Crawler’s ETL Process in this project is simple as it only combines the columns ’Edited, Spoiler, Stickied’ into one column, and then drops the previous columns as it will have no use. After completing the internal ETL Process, the transformed dataframe will be loaded to our S3 bucket.

Running the Crawler will result in the following UI:



Crawler successfully starting
The following crawler is now starting: "reddit_crawler"

reddit_crawler

Last updated (UTC)
February 15, 2025 at 10:35:28

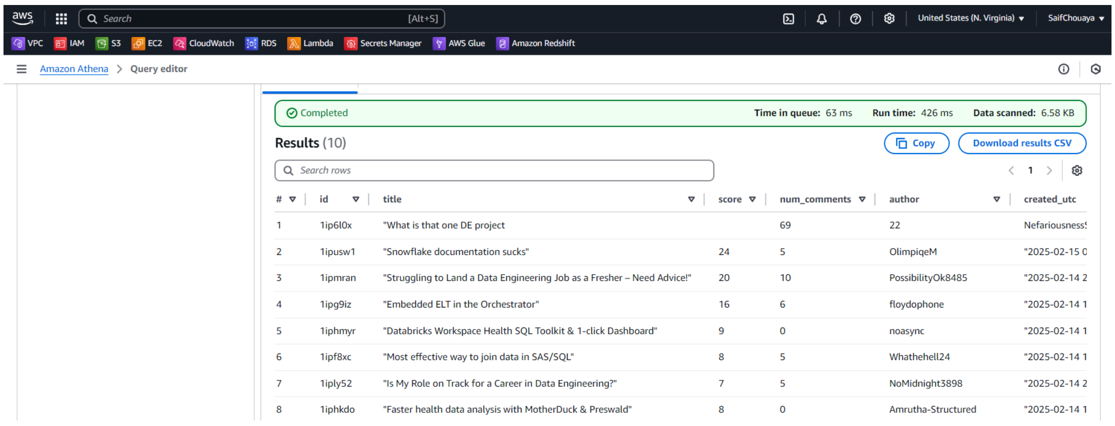
Crawler properties

Name	IAM role	Database	State
reddit_crawler	AWSGlueServiceRole-reddit-glue-role-	reddit_db_moetez	READY

Figure 6: Successfully ran and started Crawler in AWS Glue

After obtaining our transformed data from AWS Glue and storing it in our S3 bucket, AWS Athena will allow us to run SQL queries directly on S3-stored data.

The benefit of Athena in this case is that it allows for EDA and Data Visualization without the need to load the transformed data into a database.


Completed Time in queue: 63 ms Run time: 426 ms Data scanned: 6.58 KB

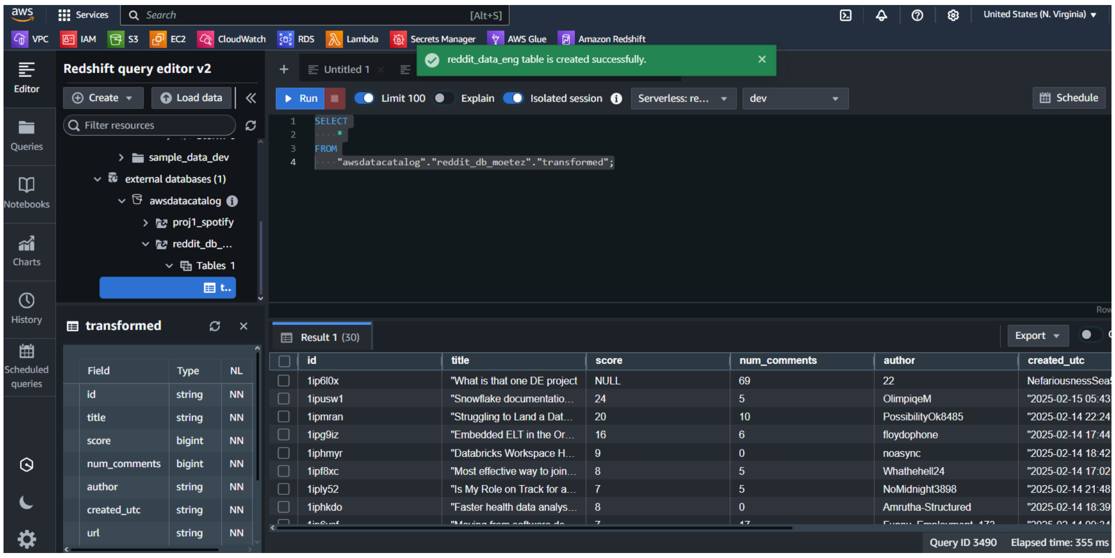
Results (10)

#	id	title	score	num_comments	author	created_utc
1	1ip6l0x	"What is that one DE project		69	22	Nefariousness
2	1ipusw1	"Snowflake documentation sucks"	24	5	OlimpiqeM	"2025-02-15 0
3	1ipmran	"Struggling to Land a Data Engineering Job as a Fresher – Need Advice!"	20	10	PossibilityOk8485	"2025-02-14 2
4	1ipg9iz	"Embedded ELT in the Orchestrator"	16	6	floydophone	"2025-02-14 1
5	1iphmyr	"Databricks Workspace Health SQL Toolkit & 1-click Dashboard"	9	0	noasync	"2025-02-14 1
6	1ipf8xc	"Most effective way to join data in SAS/SQL"	8	5	Whathehell24	"2025-02-14 1
7	1iply52	"Is My Role on Track for a Career in Data Engineering?"	7	5	NoMidnight3898	"2025-02-14 2
8	1iphkdo	"Faster health data analysis with MotherDuck & Preswald"	8	0	Amrutha-Structured	"2025-02-14 1

Figure 7: AWS Athena UI

Unlike AWS Athena, we start by loading the structured dataframe from S3 bucket into AWS Redshift, which will ensure efficient querying and will allow us to make dashboards and analytics on our dataframe.

The purpose of Redshift in this project's case is very similar to Athena, but it still allows us to perform more complex queries and aggregations that will derive great insights from our data.


Redshift query editor v2

reddit_data_eng table is created successfully.

Limit 100 · Explain · Isolated session · Serverless: re... · dev

```
SELECT
    *
FROM
    "awsdatacatalog"."reddit_db_moetez"."transformed";
```

transformed

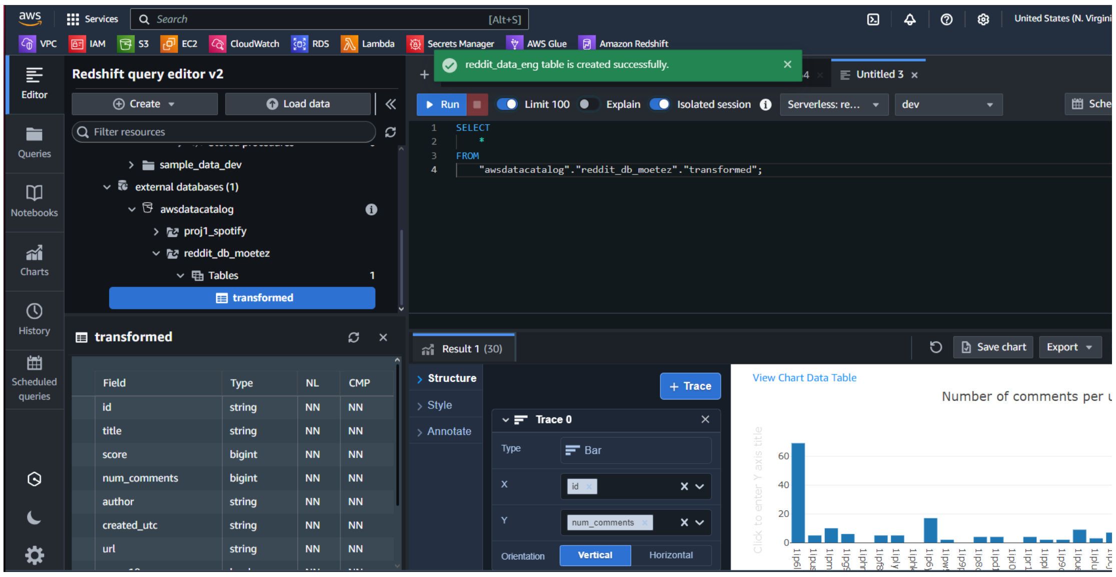
Field	Type	NL
id	string	NN
title	string	NN
score	bigint	NN
num_comments	bigint	NN
author	string	NN
created_utc	string	NN
url	string	NN

Result 1 (30)

id	title	score	num_comments	author	created_utc
1ip6l0x	"What is that one DE project	NULL	69	22	NefariousnessSea
1ipusw1	"Snowflake documentatio...	24	5	OlimpiqeM	"2025-02-15 05:43
1ipmran	"Struggling to Land a Dat...	20	10	PossibilityOk8485	"2025-02-14 22:24
1ipg9iz	"Embedded ELT in the Or...	16	6	floydophone	"2025-02-14 17:44
1iphmyr	"Databricks Workspace H...	9	0	noasync	"2025-02-14 18:42
1ipf8xc	"Most effective way to join...	8	5	Whathehell24	"2025-02-14 17:02
1iply52	"Is My Role on Track for a...	7	5	NoMidnight3898	"2025-02-14 21:48
1iphkdo	"Faster health data analys...	8	0	Amrutha-Structured	"2025-02-14 18:39

Query ID 3490 Elapsed time: 355 ms

Figure 8: Data Warehousing with AWS Redshift



reddit_data_eng table is created successfully.

```
SELECT
    *
FROM
    "awsdatacatalog"."reddit_db_moetez"."transformed";
```

transformed

Field	Type	NL	CMP
id	string	NN	NN
title	string	NN	NN
score	bigint	NN	NN
num_comments	bigint	NN	NN
author	string	NN	NN
created_utc	string	NN	NN
url	string	NN	NN

Figure 9: Data Visualization in AWS Redshift

CHAPTER 5 Conclusion

This project successfully demonstrated the end-to-end process of building a data pipeline using Reddit API, Airflow, Celery, Postgres, Docker, Redis, S3, AWS Glue, Athena, and Redshift. It automates the process of Extraction of a subreddit data, Transforming and Manipulating the data with AWS Glue, and Loading Reddit's subreddit data to AWS Athena and Redshift while ensuring efficient querying and visualization through Amazon Web Services.
After querying our data with Redshift, we can take this project further by exporting the data to another AWS visualization service or any other Data Visualization platform (aka. Power BI, Tableau or IBM Cognos Analytics) to create a professional dashboard and derive insights from our data. This part is not included in this project since our main goal is to create data pipelines and automate the ETL process of Reddit's Data.
The project still needs several future enhancements, including real-time streaming, Machine Learning integration and extended data analysis. These enhancements will be applied on this project in the near future with other data tools.