[TYPE THE COMPANY NAME]

Reddit Comments ETL Pipeline

UsingAWS and Airflow

Dhanyamol Devassy

Table of Contents

1. Introduction

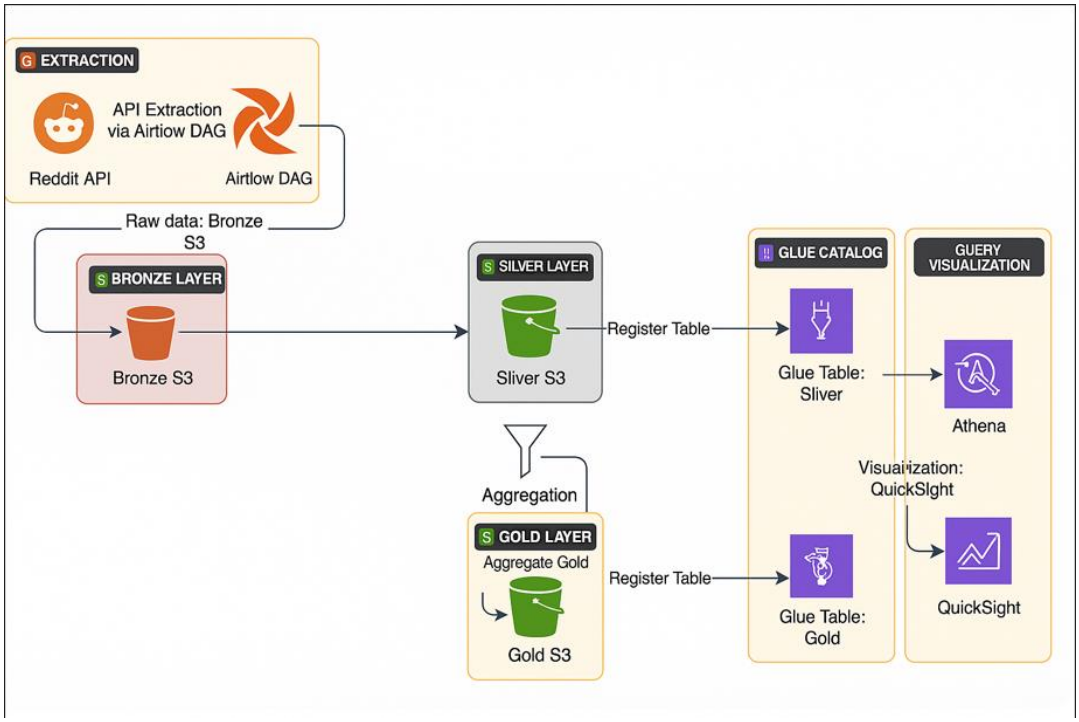
In the era of big data, Reddit represents a valuable platform for analyzing public conversations. With millions of daily active users, Reddit offers real-time insights into community opinions, discussions, and sentiment across domains like Python, Machine Learning, AWS, and Data Science.

This project implements a cloud-native data pipeline for Reddit comments using AWS services (S3, Glue, Athena, QuickSight) orchestrated with Apache Airflow. It follows the medallion architecture (Bronze → Silver → Gold) for structured ETL and produces business-ready dashboards in QuickSight.

The pipeline automatically:

1. Extracts Reddit comments using the Reddit API.
2. Stores raw data in Amazon S3 (Bronze).
3. Cleans & normalizes data via Glue ETL jobs (Silver).
4. Aggregates & enriches subreddit metrics (Gold).
5. Validates & queries datasets using Athena.
6. Visualizes insights with QuickSight dashboards.

2. Architecture Diagram



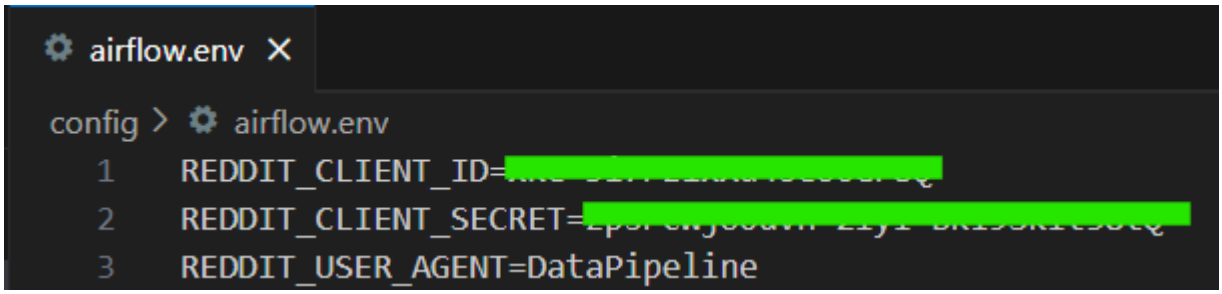
3. Data Source: Reddit API

3.1 Why Reddit API?

Reddit provides an official API for fetching posts, comments, votes, and metadata from subreddits. For this project, the comment stream was chosen since it represents direct user interaction and sentiment.

3.2 API Authentication

The pipeline uses OAuth2-based credentials stored in Airflow .env:



3.3 Helper Script (reddit_api.py)

The script fetches comments from multiple subreddits and saves them to CSV for downstream ingestion.

reddit_api.py 1

utils > reddit_api.py > fetch_reddit_comments

```python
import os
import praw
import pandas as pd

def fetch_reddit_comments(output_path="data/raw/reddit_raw.csv"):
    reddit = praw.Reddit(
        client_id=os.getenv("CLIENT_ID"),
        client_secret=os.getenv("CLIENT_SECRET"),
        user_agent=os.getenv("USER_AGENT")
    )

    subreddits = ["datascience", "python", "aws", "machinelearning"]
    comments = []

    for sub in subreddits:
        for comment in reddit.subreddit(sub).comments(limit=500):
            comments.append({
                "id": comment.id,
                "author": str(comment.author),
                "subreddit": comment.subreddit.display_name,
                "body": comment.body,
                "created_utc": comment.created_utc
            })

    df = pd.DataFrame(comments)
    os.makedirs(os.path.dirname(output_path), exist_ok=True)
    df.to_csv(output_path, index=False)
    print(f"Saved {len(df)} comments to {output_path}")

```

3.4 Data Collected

- Subreddit name
- Comment ID
- Comment text
- Timestamp (created_utc)
- Derived sentiment score

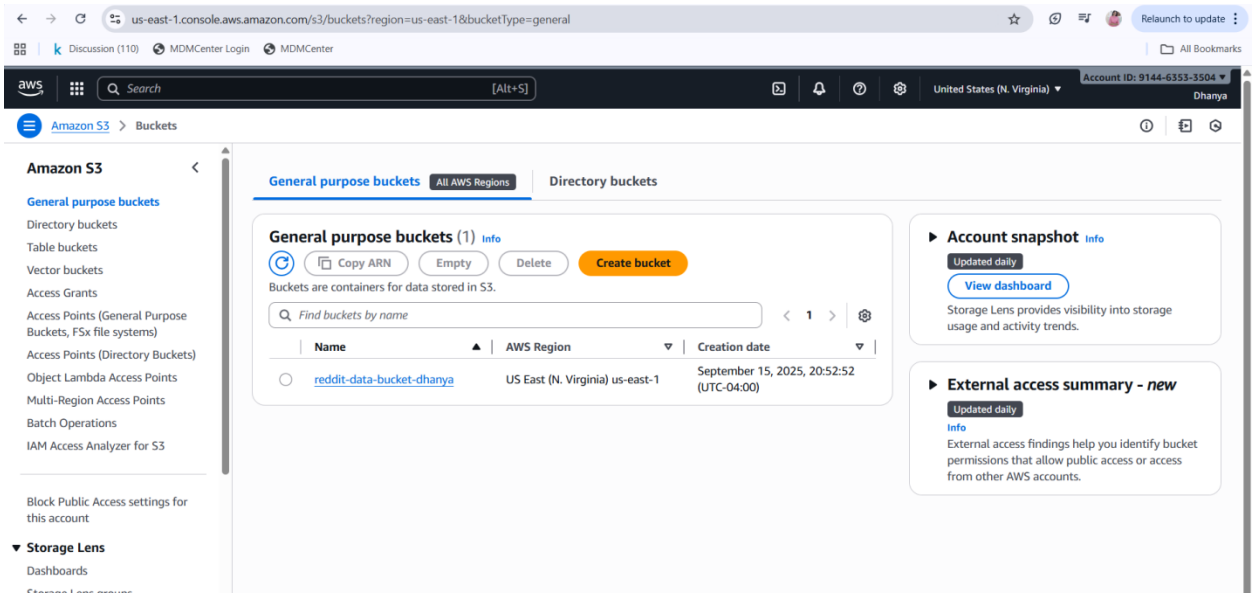
The Python script reddit_api.py fetches data and stores daily dumps into data/raw/reddit_raw.csv.

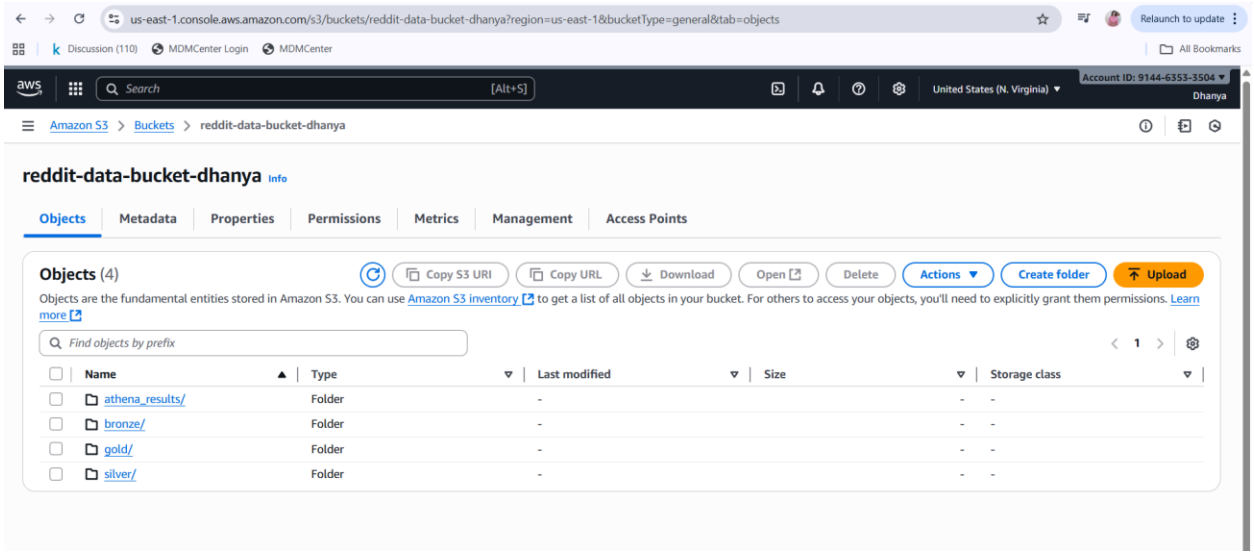
Output: data/raw/reddit_raw.csv (daily).

4. Storage: Amazon S3 (Data Lake)

The project follows partitioned storage for scalability.

```
C:\Users\Owner>aws s3 mb s3://reddit-data-bucket-dhanya --region us-east-1
make_bucket: reddit-data-bucket-dhanya

C:\Users\Owner>
```



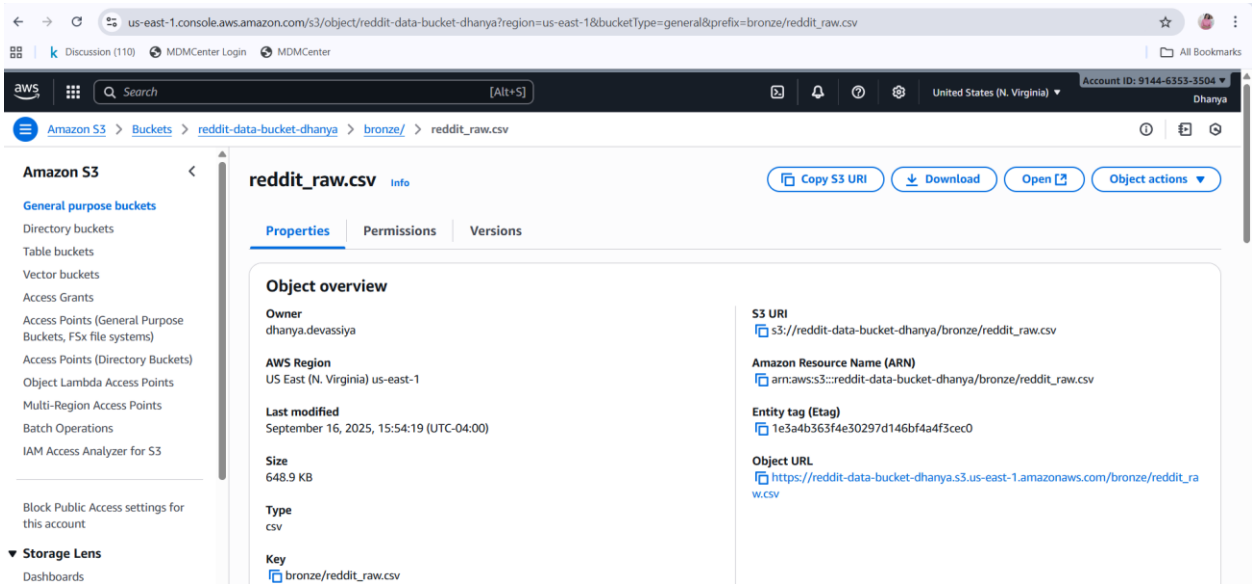


4.1 Bronze Layer

- Raw CSV dumps, stored daily.
- Path format:

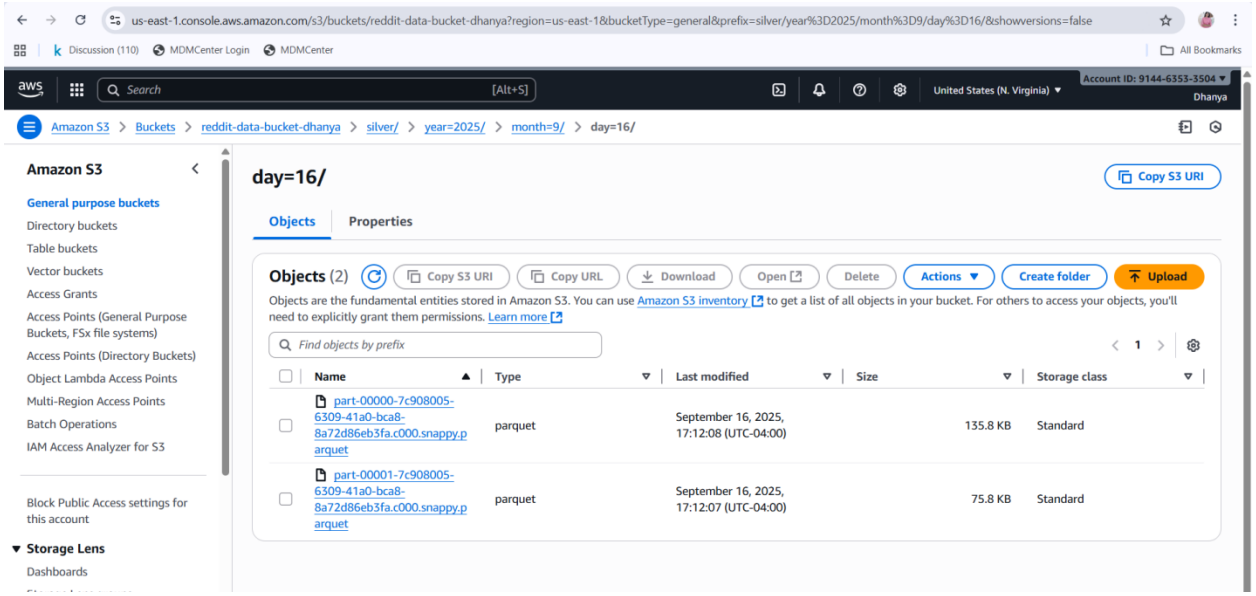
s3://reddit-data-bucket-dhanya/reddit/bronze/reddit_raw.csv

Purpose: Preserve original Reddit API data for traceability



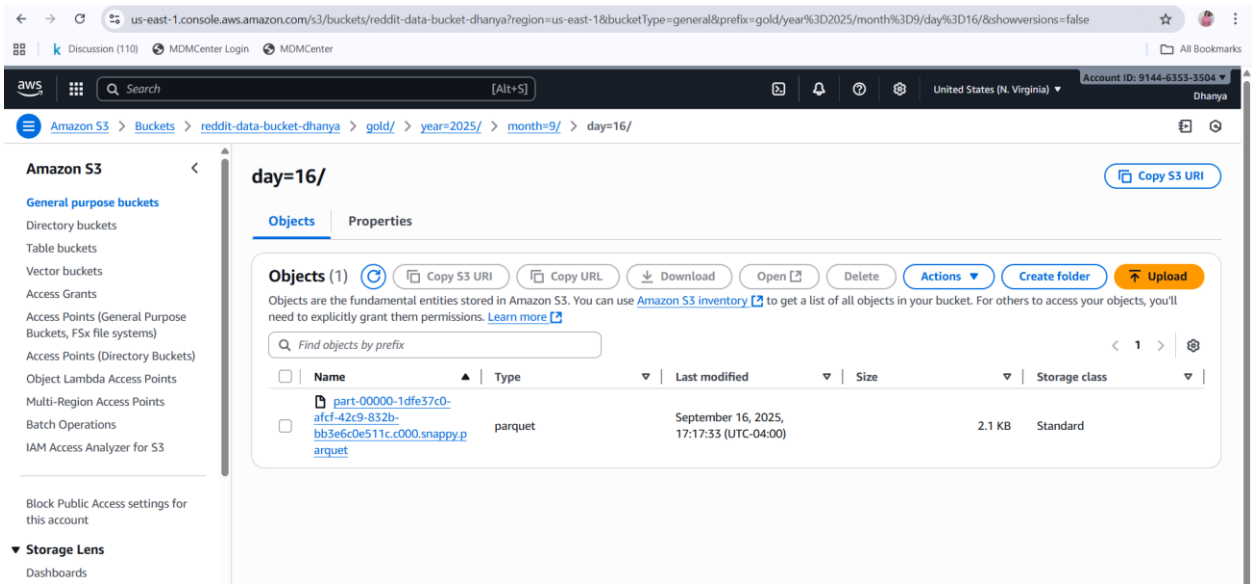
4.2 Silver Layer

- ETL job cleans raw CSVs:
  - Extracts required fields (subreddit, comment, sentiment).
  - Removes malformed/empty records.
  - Normalizes timestamps into year, month, day.
- Stored in Parquet format for query efficiency.



4.3 Gold Layer

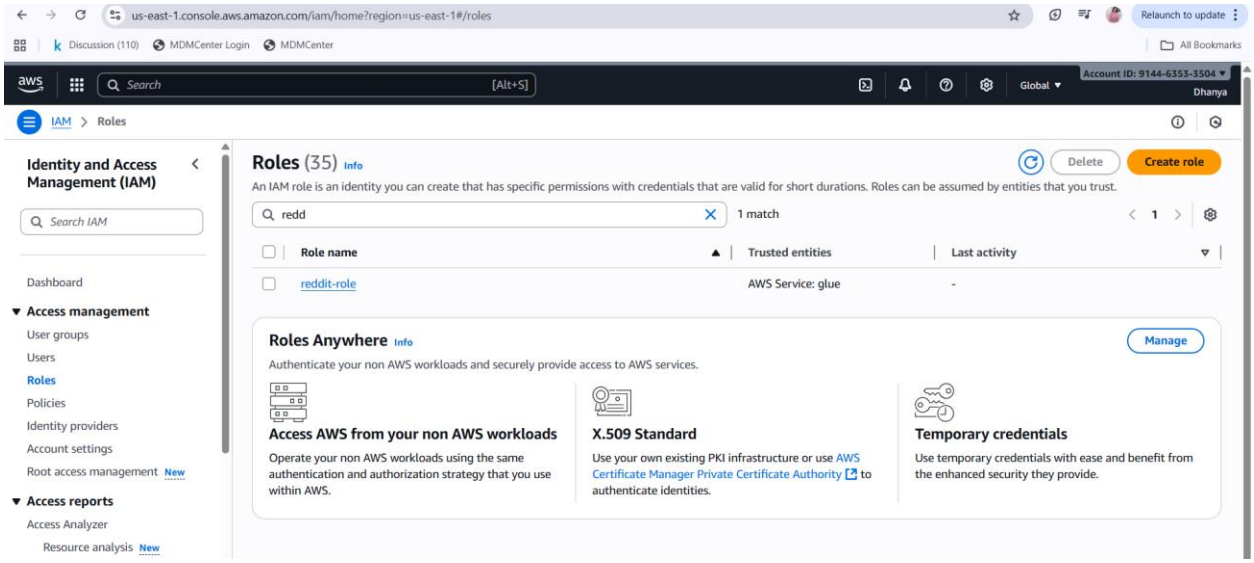
- Aggregation layer:
  - Groups by subreddit.
  - Computes comment_count and avg_sentiment.
- Stored in Parquet, partitioned by year/month/day.
- Directly queried by Athena and used in QuickSight.



5. AWS Glue Resources

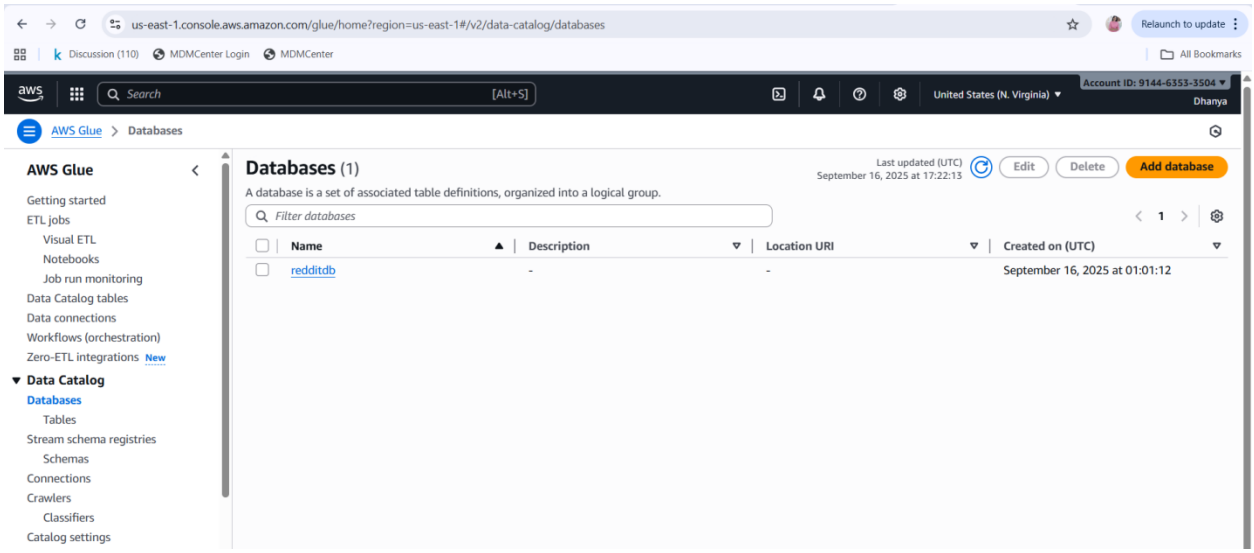
5.1 IAM Role for Glue

- Role Name: reddit-role.
- Permissions:
    - AmazonS3FullAccess (to dataset bucket).
    - AWSGlueServiceRole managed policy.
    - AmazonAthenaFullAccess (optional for validation).



5.2 Create a Glue Database

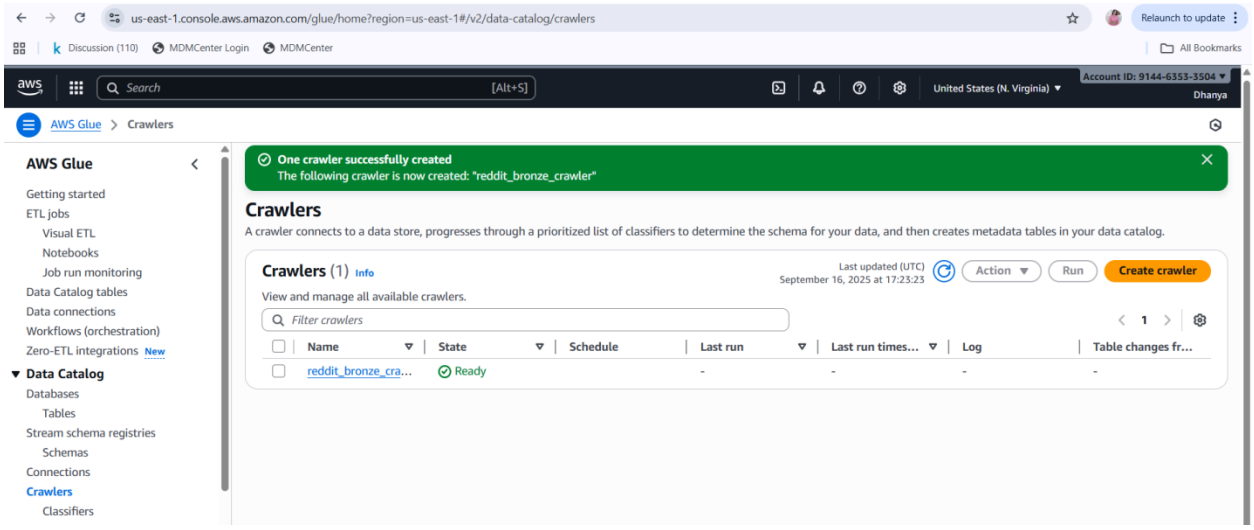
1. Go to AWS Glue Console → Databases → Add Database.
2. Name: redditdb.
3. Used as metadata catalog for Athena & QuickSight.



5.3 Create Glue Crawlers

Crawler for Bronze Layer

- Name: reddit-bronze-crawler.
- Data store: s3://reddit-data-dhanya/reddit/bronze/.
- IAM Role: reddit-role.
- Target Database: redditdb.
- Schedule: On-demand (triggered via Airflow).



Ran the crawler

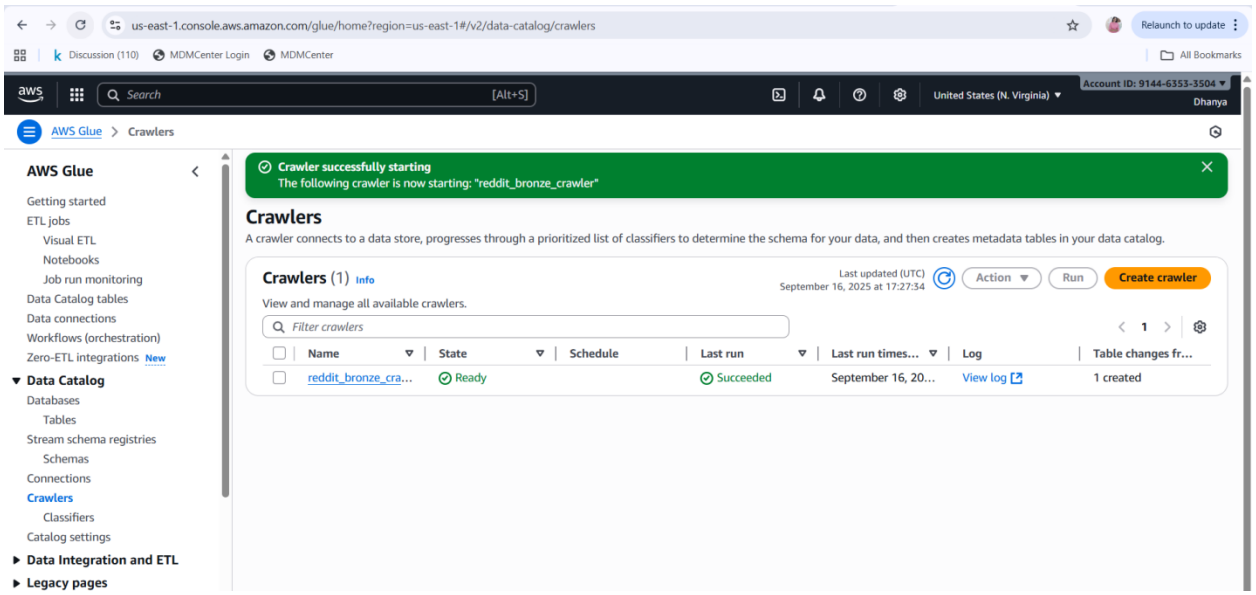
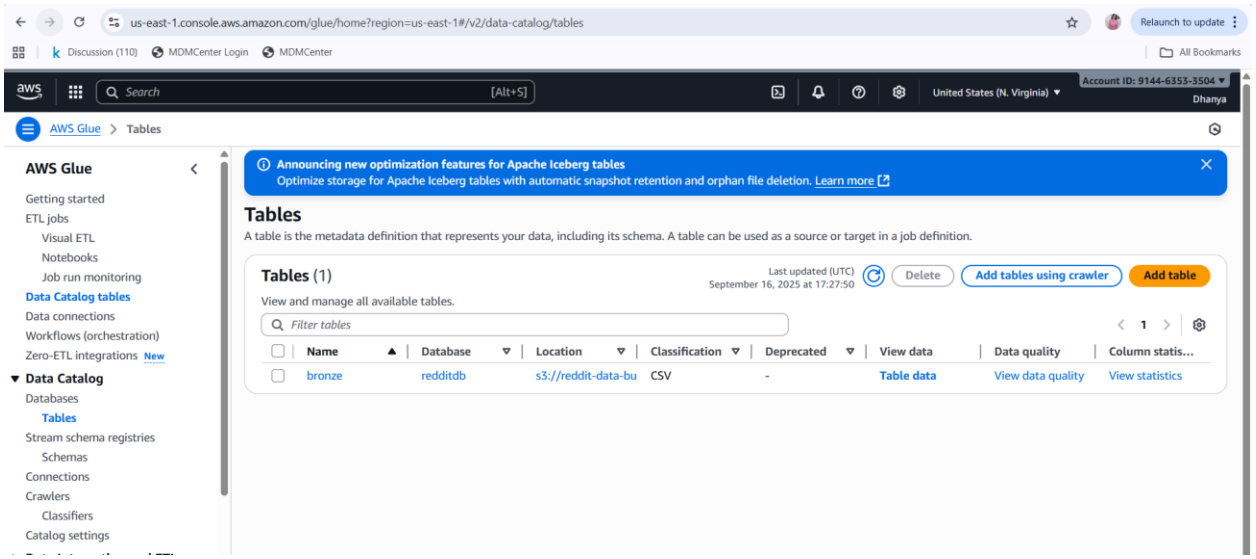
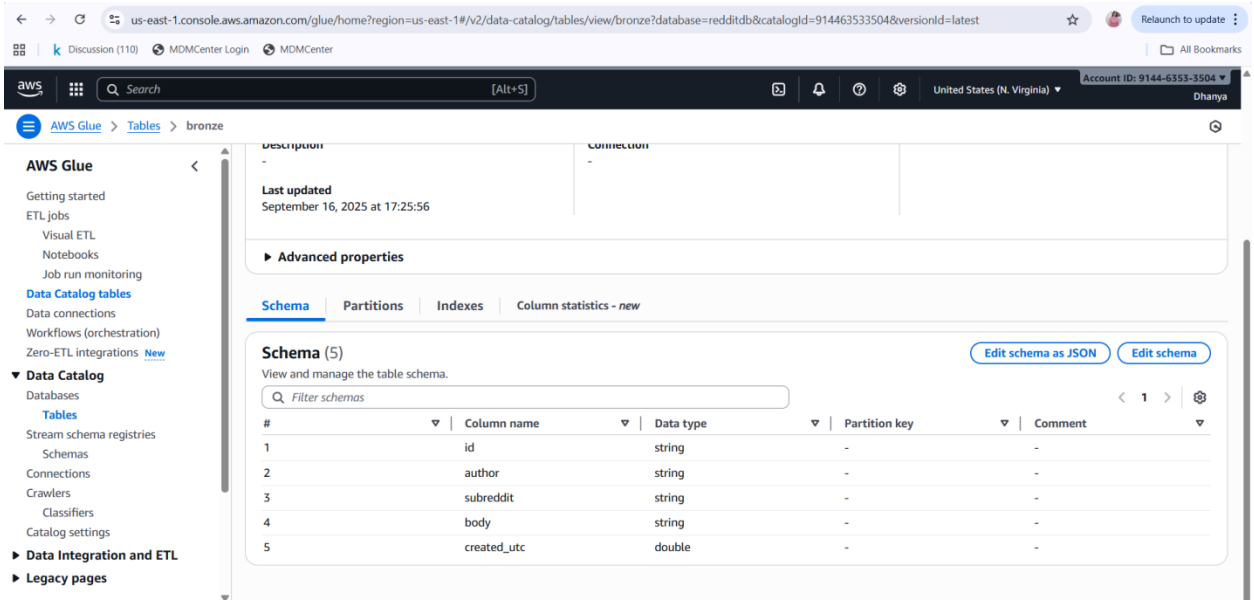
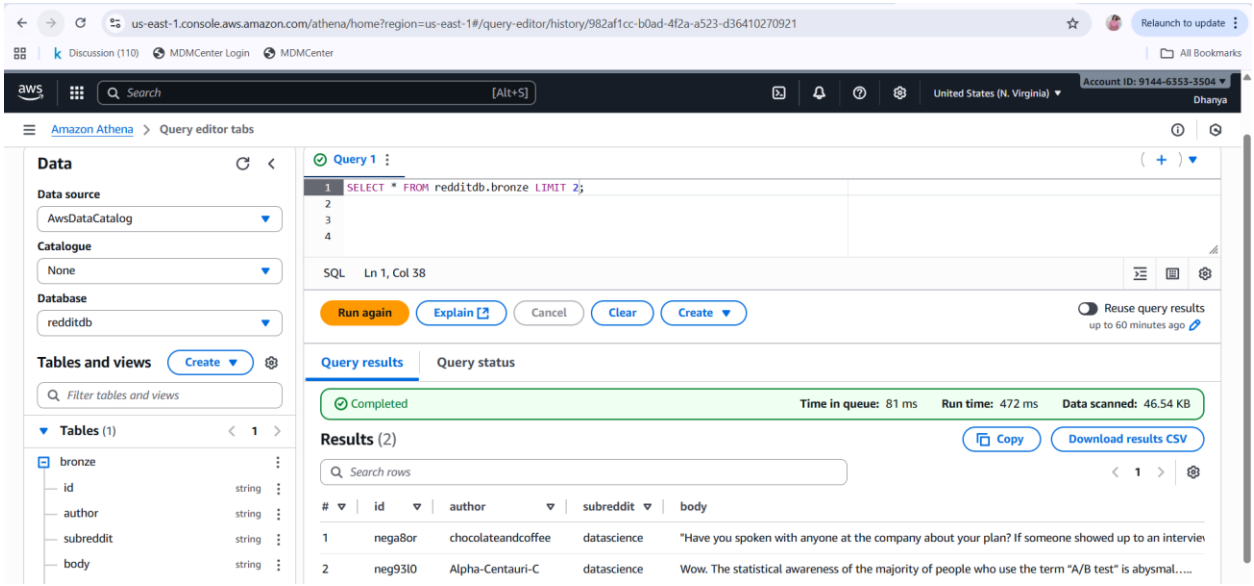


Table is created in redditdb





Test in Athena



5.4 Create Glue Jobs

5.4.1 Silver Job (silver-job)

- Input: Bronze CSV.
- Transformations:
    - Select key columns (subreddit, text, sentiment, timestamp).

- Add partitions: year, month, day.
- Save as Parquet.

Script:

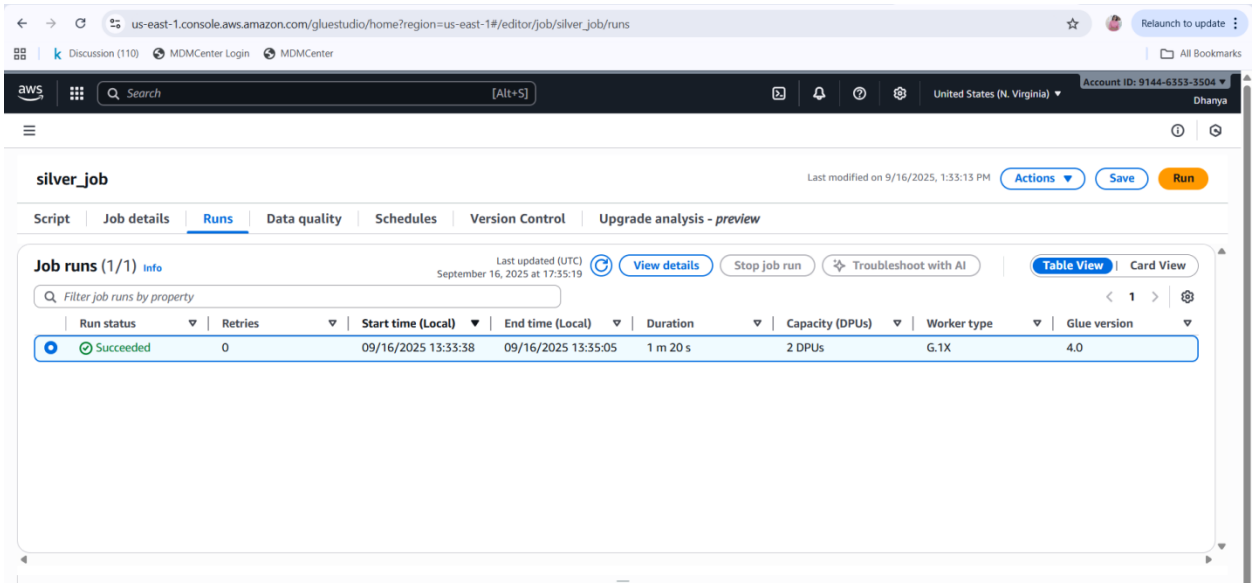
glue_silver_job.py 1

etls > glue_silver_job.py > ...

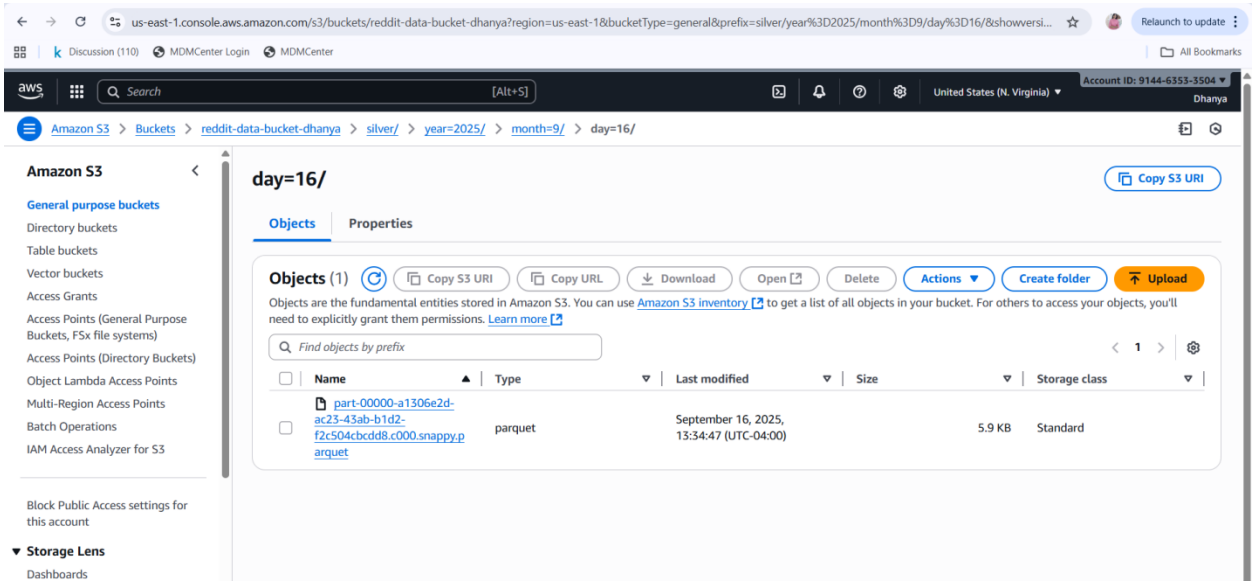
```python
from awsglue.context import GlueContext
from pyspark.context import SparkContext
from pyspark.sql.functions import col, from_unixtime, year, month, dayofmonth

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

# Read Bronze (CSV)
df = spark.read.option("header", True).csv("s3://reddit-data-bucket-dhanya/bronze/")

# Clean + add partition columns
df_clean = (
    df.filter(col("body").isNotNull())
      .withColumn("created_ts", from_unixtime(col("created_utc")))
      .withColumn("year", year(from_unixtime("created_utc")))
      .withColumn("month", month(from_unixtime("created_utc")))
      .withColumn("day", dayofmonth(from_unixtime("created_utc")))
)

# Write to Silver as partitioned Parquet
df_clean.write.mode("overwrite").partitionBy("year","month","day") \
    .parquet("s3://reddit-data-bucket-dhanya/silver/")

```

Run the job

Reddit Comments AWS Data Pipeline



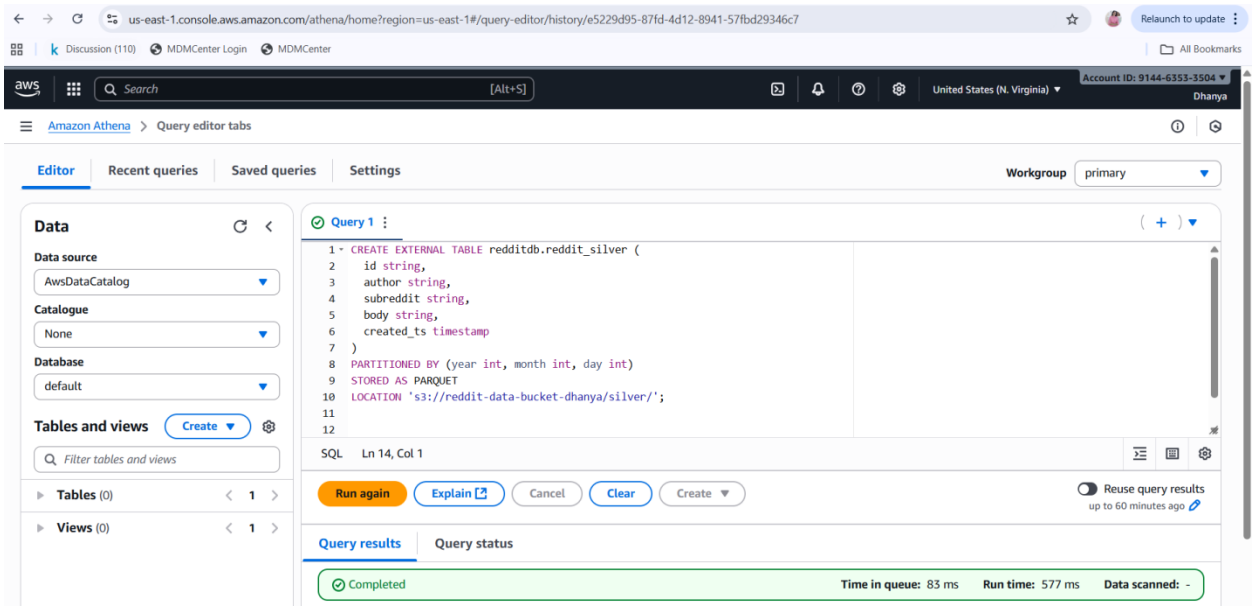
Output is generated in s3 with partitions



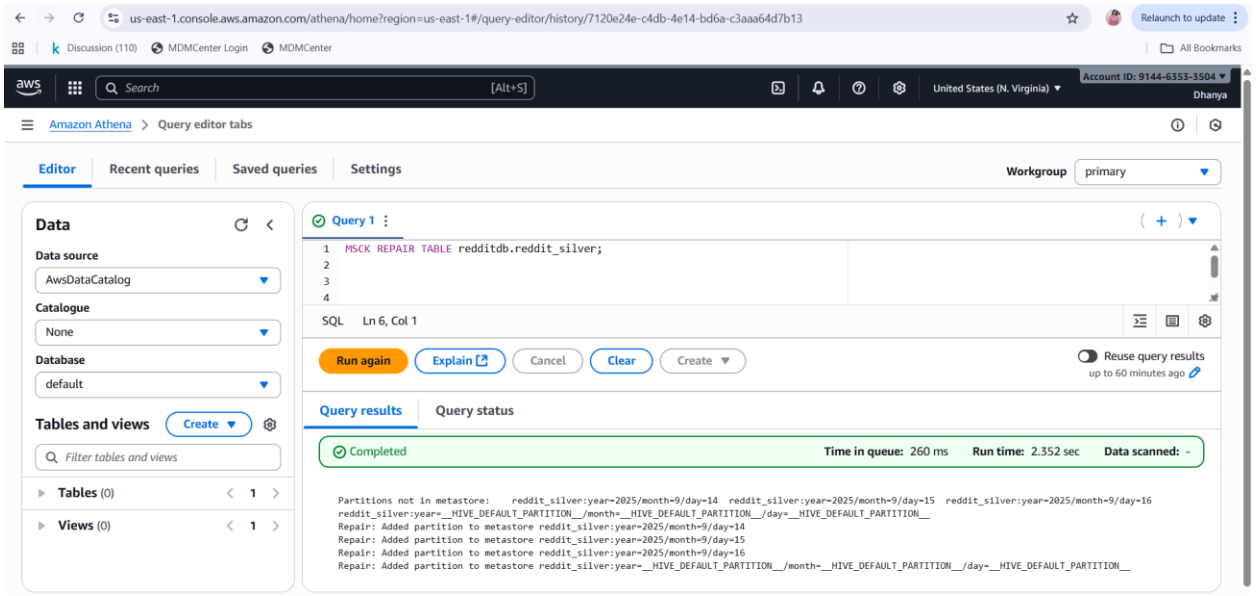
Validate in Athena

Create Silver Table

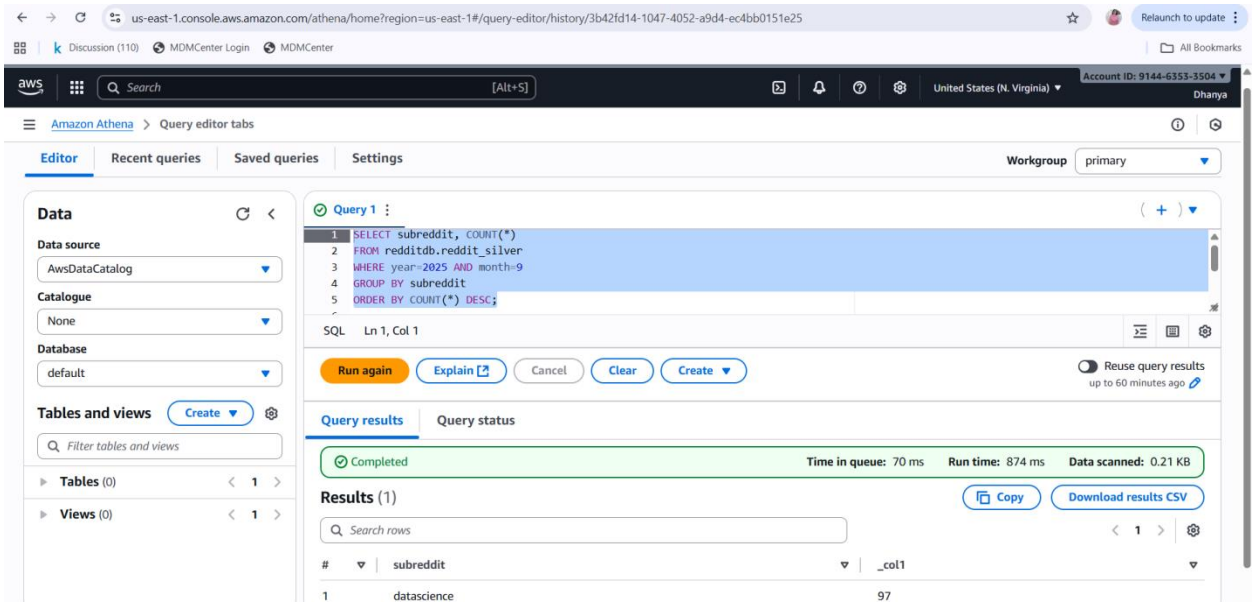
Reddit Comments AWS Data Pipeline



Refresh partitions



Test query



5.4.2 Gold Job (gold-job)

- Input: Silver dataset.
- Transformations:
  - Group by subreddit.
  - Calculate comment_count and avg_sentiment.

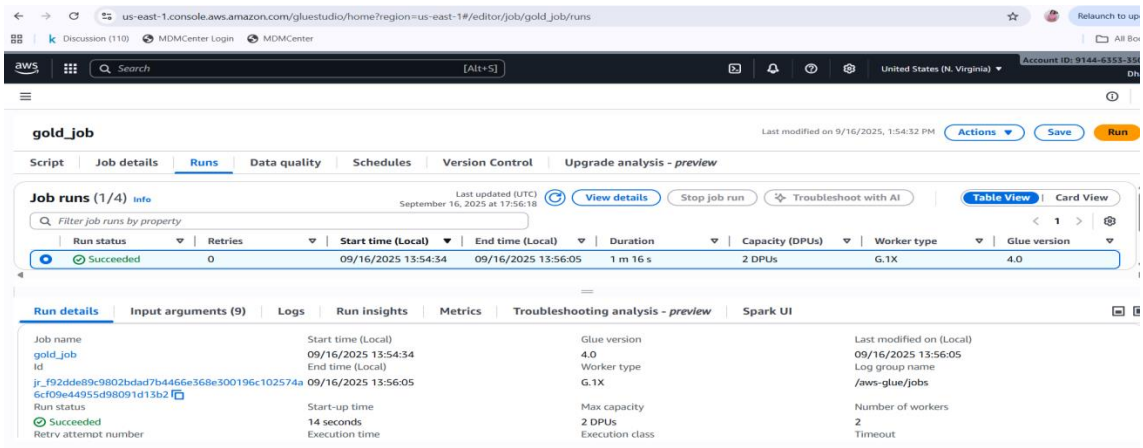
Script:

glue_gold_job.py 1

etls > glue_gold_job.py > ...

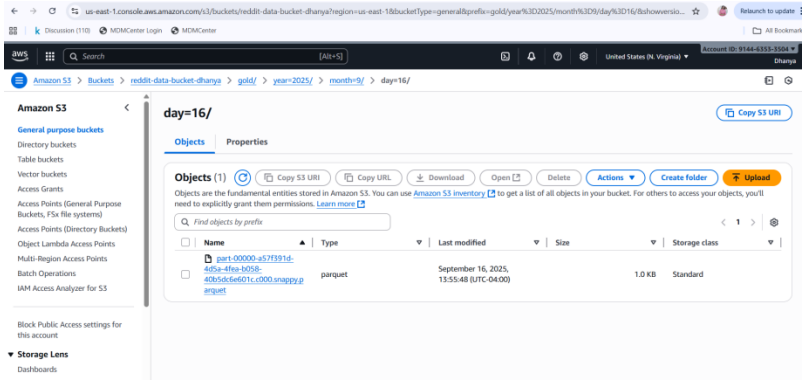
```python
from awsglue.context import GlueContext
from pyspark.context import SparkContext
from pyspark.sql.functions import col, count, avg, udf
from pyspark.sql.types import DoubleType
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
# Small lexicon
positive_words = {"good", "great", "excellent", "happy", "love", "like", "awesome"}
negative_words = {"bad", "terrible", "sad", "hate", "angry", "worst", "awful"}
def sentiment_score(text):
    if not text:
        return 0.0
    words = text.lower().split()
    score = 0
    for w in words:
        if w in positive_words:
            score += 1
        elif w in negative_words:
            score -= 1
    return float(score)
sentiment_udf = udf(sentiment_score, DoubleType())
# Read Silver
df = spark.read.parquet("s3://reddit-data-bucket-dhanya/silver/")
# Add sentiment
df_sent = df.withColumn("sentiment", sentiment_udf(col("body")))
# Aggregate into Gold
df_gold = (
    df_sent.groupBy("subreddit","year","month","day")
           .agg(
               count("id").alias("comment_count"),
               avg("sentiment").alias("avg_sentiment")
           )
)
df_gold.write.mode("overwrite").partitionBy("year","month","day") \
    .parquet("s3://reddit-data-bucket-dhanya/gold/")

```

Run the job



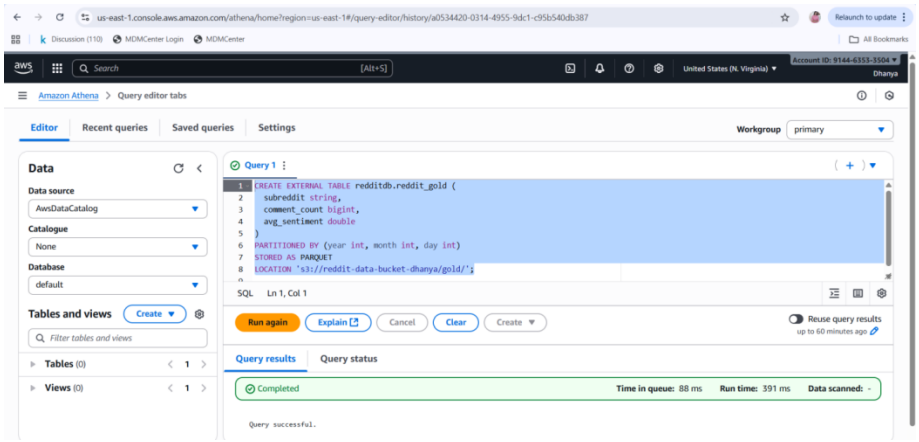
Reddit Comments AWS Data Pipeline

Output is generated in s3

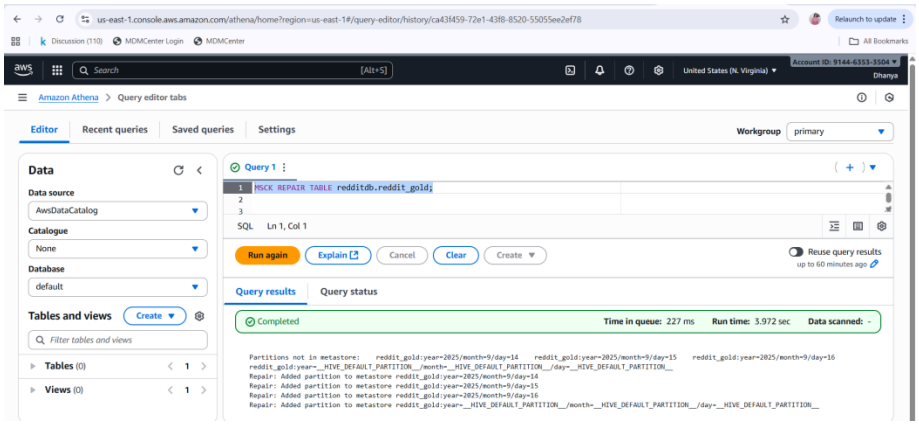


Validate in Athena

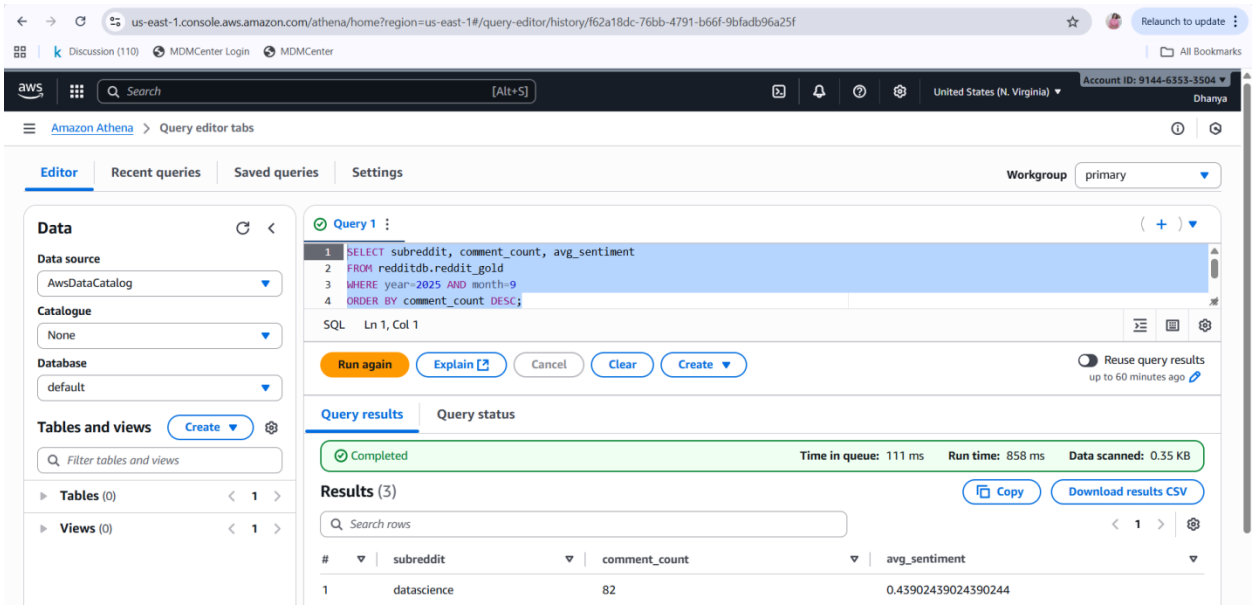
Create gold table



Refresh partitions



Test query



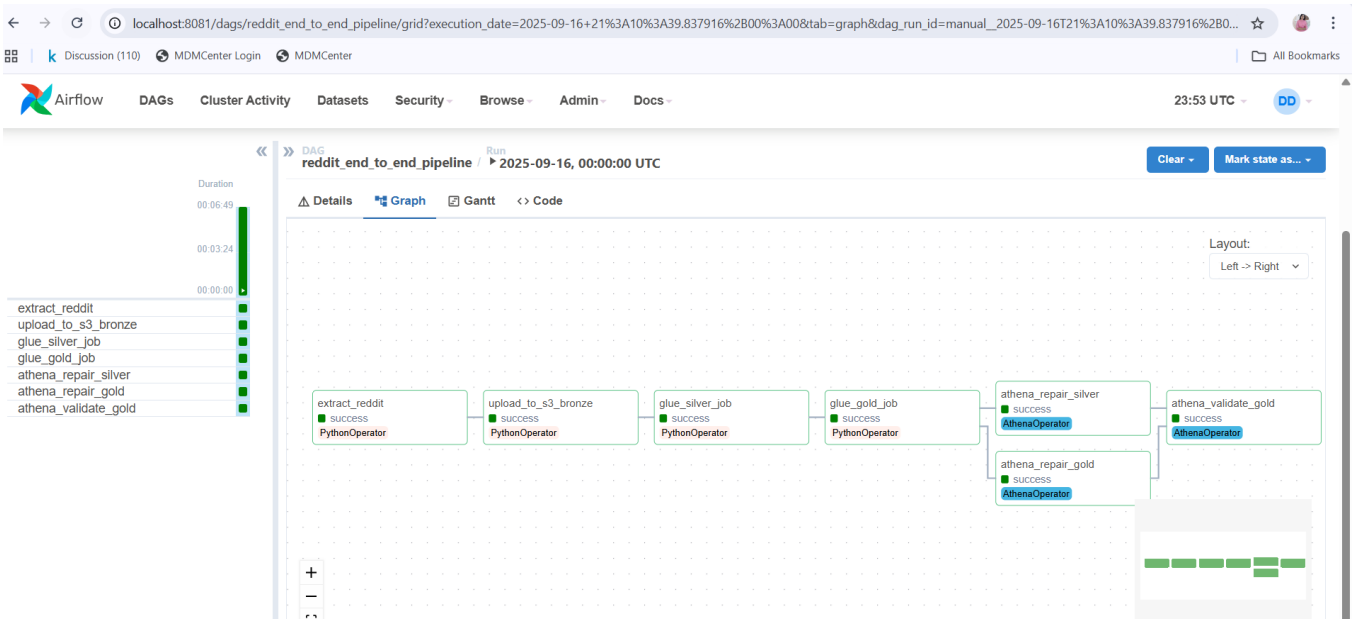
6. Airflow Orchestration

The DAG in Airflow (reddit_dag.py) runs in sequence:

1. extract_reddit → Pulls data.
2. upload_to_s3 → Push Bronze to S3.
3. trigger_glue_silver → Cleans data.
4. trigger_glue_gold → Aggregates metrics.
5. athena_repair → Updates partitions.
6. validate_data → Ensures non-empty gold tables.

Improvements made:

- .env variables for flexibility.
- Boto3 start_job_run for Glue reliability.
- Partition repair with Athena.



7. Athena Validation Queries (Samples)

-- Repair tables

MSCK REPAIR TABLE redditdb.reddit_silver;

MSCK REPAIR TABLE redditdb.reddit_gold;

-- Top subreddits by comments

SELECT subreddit, SUM(comment_count)

FROM redditdb.reddit_gold

GROUP BY subreddit

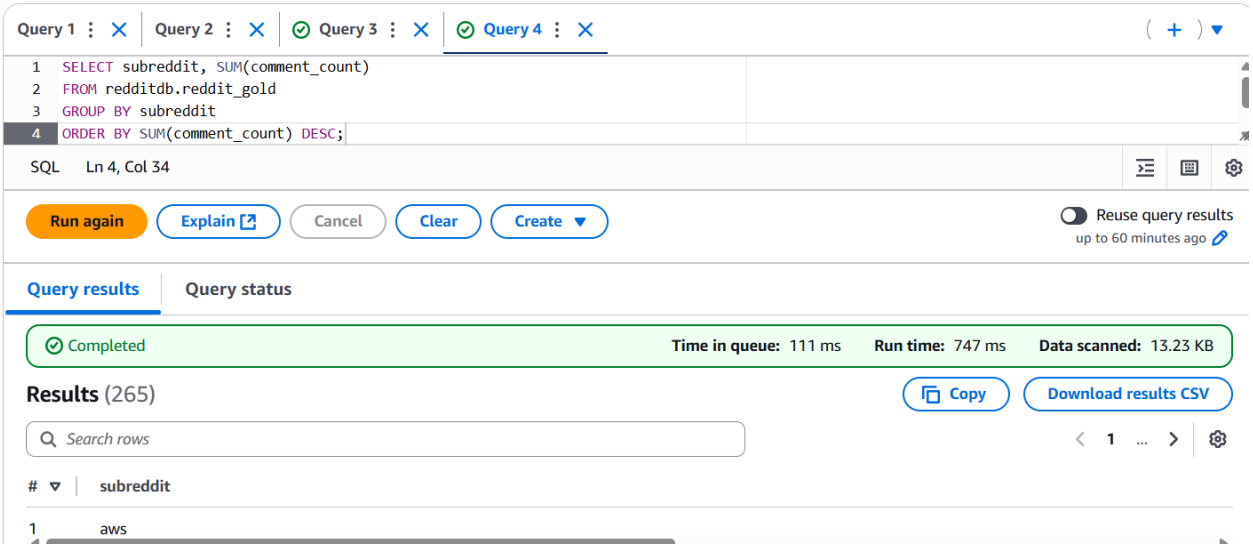
ORDER BY SUM(comment_count) DESC;

-- Average sentiment

SELECT subreddit, AVG(avg_sentiment)

FROM redditdb.reddit_gold

GROUP BY subreddit;



8. QuickSight Visuals

The reddit_gold dataset (output of Glue Gold jobs) was used for all visualizations. This dataset contains aggregated subreddit-level metrics:

- subreddit
- comment_count (total number of comments)
- avg_sentiment (average sentiment score per subreddit)

8.1 Top Subreddits by Comment Count (Bar Chart)

Description:

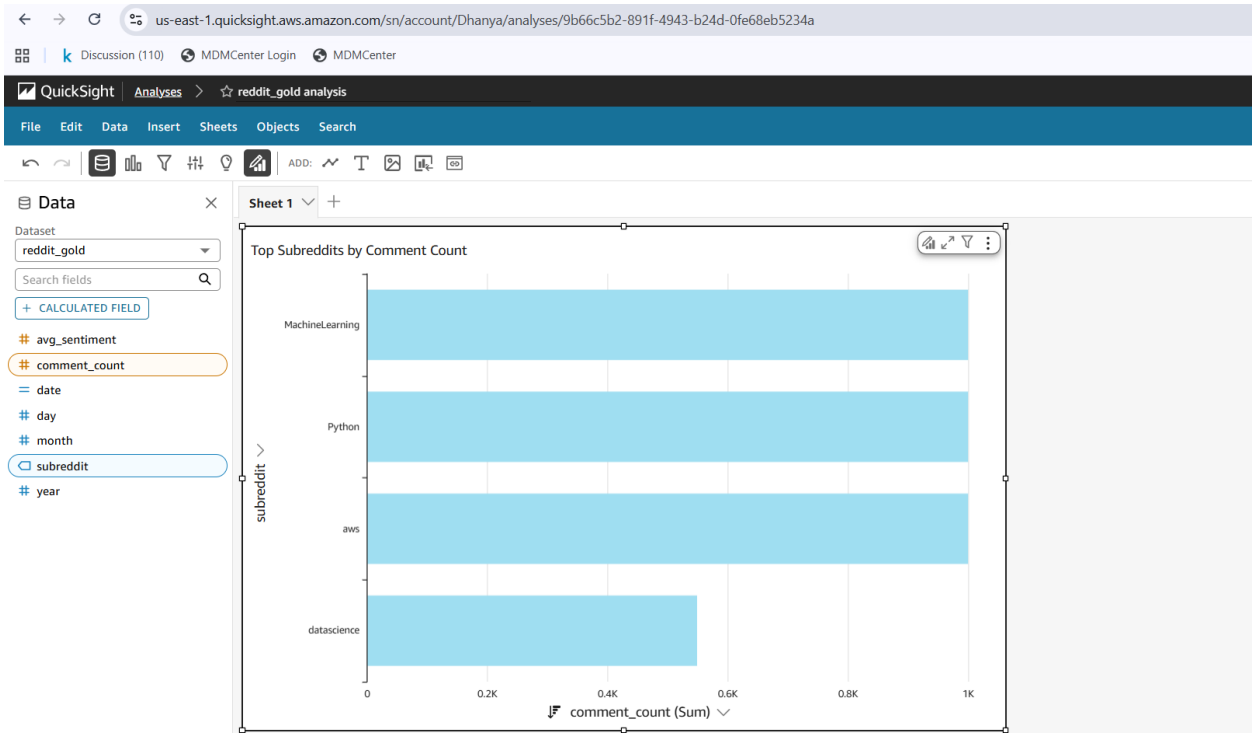
- X-axis: comment_count (Sum)
- Y-axis: subreddit
- Chart type: Horizontal bar

Purpose:

This chart highlights which subreddits are most active based on the volume of comments.

Insights:

- Subreddits like MachineLearning, Python, and AWS dominate comment activity.
- DataScience, while popular, has fewer comments compared to technical subreddits.



8.2 Average Sentiment by Subreddit (Colored Bar Chart)

Description:

- X-axis: subreddit
- Y-axis: avg_sentiment (Average)
- Color: Sentiment_Category (Negative = red, Neutral = yellow, Positive = green)

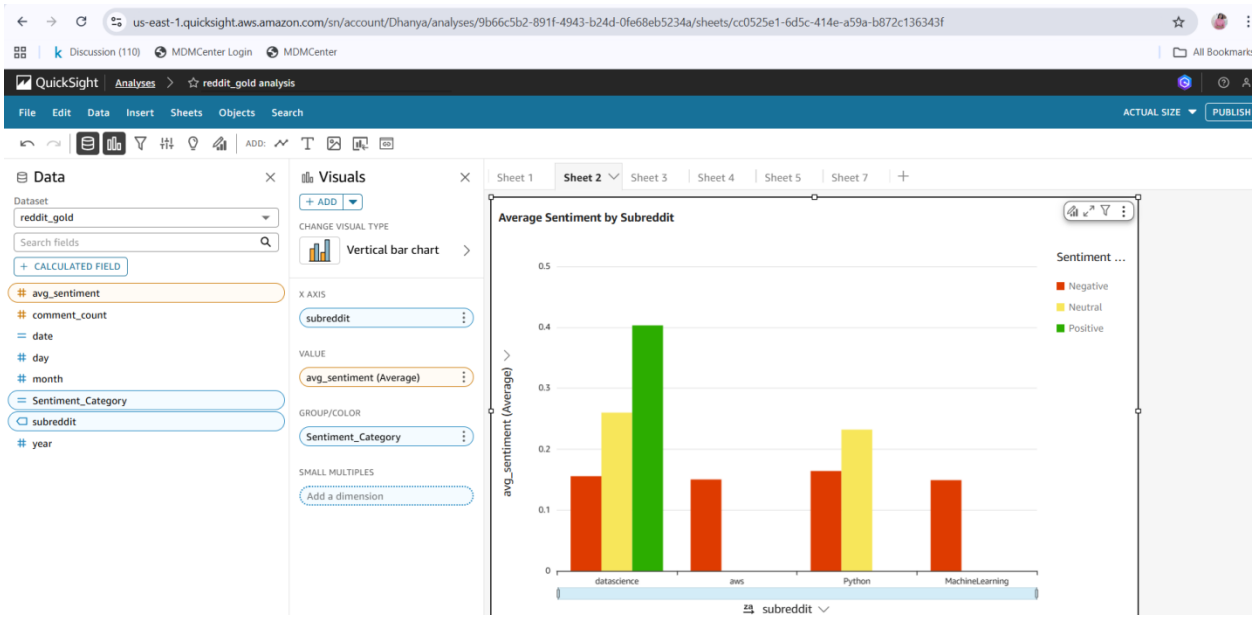
Purpose:

This chart compares the overall sentiment of discussions across subreddits.

Insights:

- DataScience subreddit shows more positivity (higher avg sentiment).

- Python and AWS communities lean neutral.
- MachineLearning has slightly more negative sentiment.
- Provides a qualitative measure of how communities feel.



8.3 Trend of Comments Over Time (Line Chart)

Description:

- X-axis: date
- Y-axis: comment_count (Sum)
- Color: subreddit

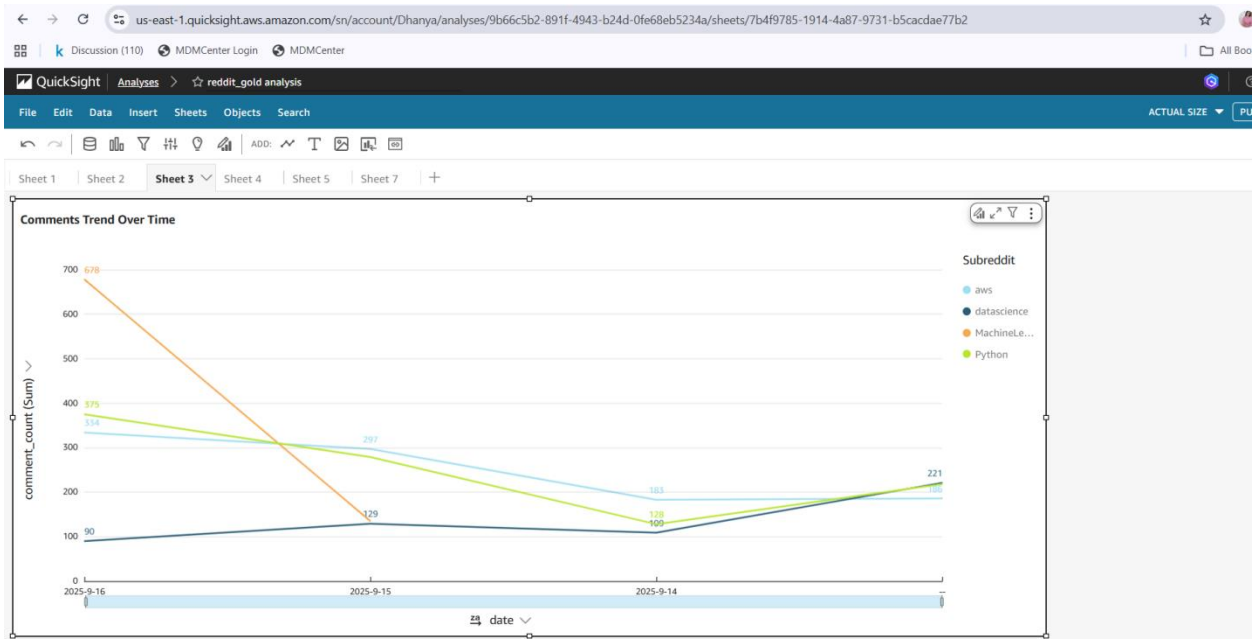
Purpose:

This chart reveals temporal patterns in activity.

Insights:

- Spikes in comments may align with events (e.g., AWS re:Invent, ML paper releases, Python updates).
- AWS subreddit shows sudden peaks and declines, suggesting event-driven activity.

- Python and MachineLearning remain steady, showing continuous community discussions.
- Useful for tracking trends and seasonality.



8.4 Sentiment Distribution (Pie Chart)

Description:

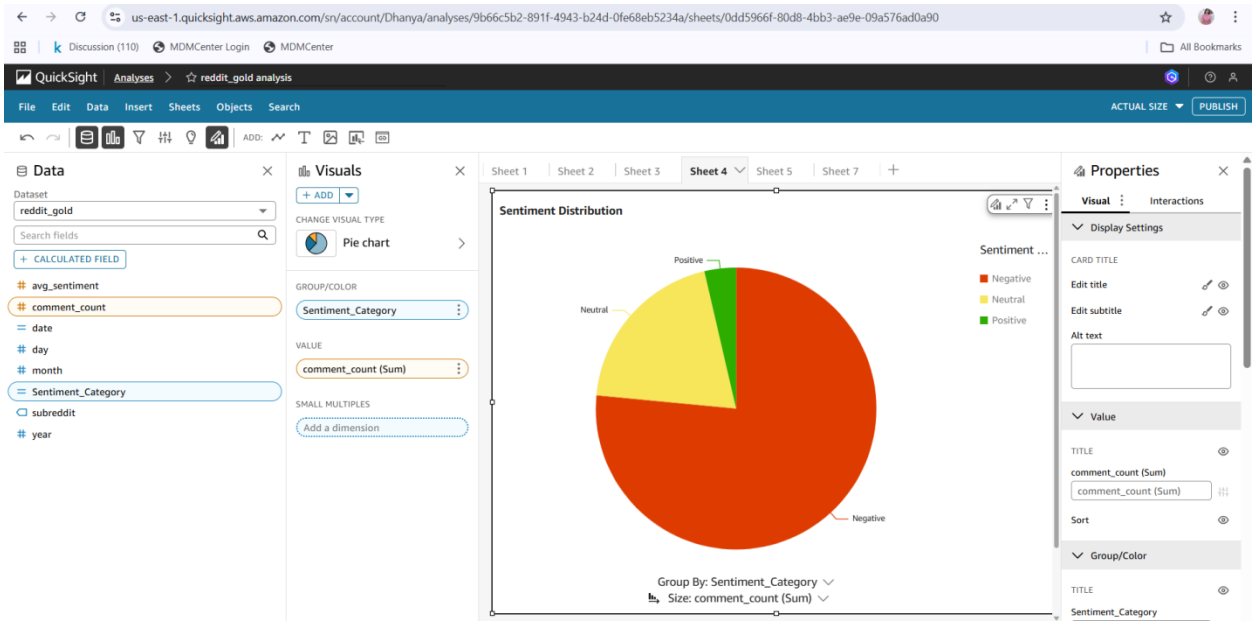
- Group/Color: Sentiment_Category
- Value: comment_count (Sum)

Purpose:

This visualization gives an overall breakdown of sentiment across all Reddit comments.

Insights:

- Majority of comments are Negative (red slice).
- Neutral (yellow) sentiment makes up a significant portion.
- Positive (green) sentiment is the smallest slice.
- Shows that Reddit discussions tend to be critical or cautious.



8.5 Heatmap (Subreddit × Day)

Description:

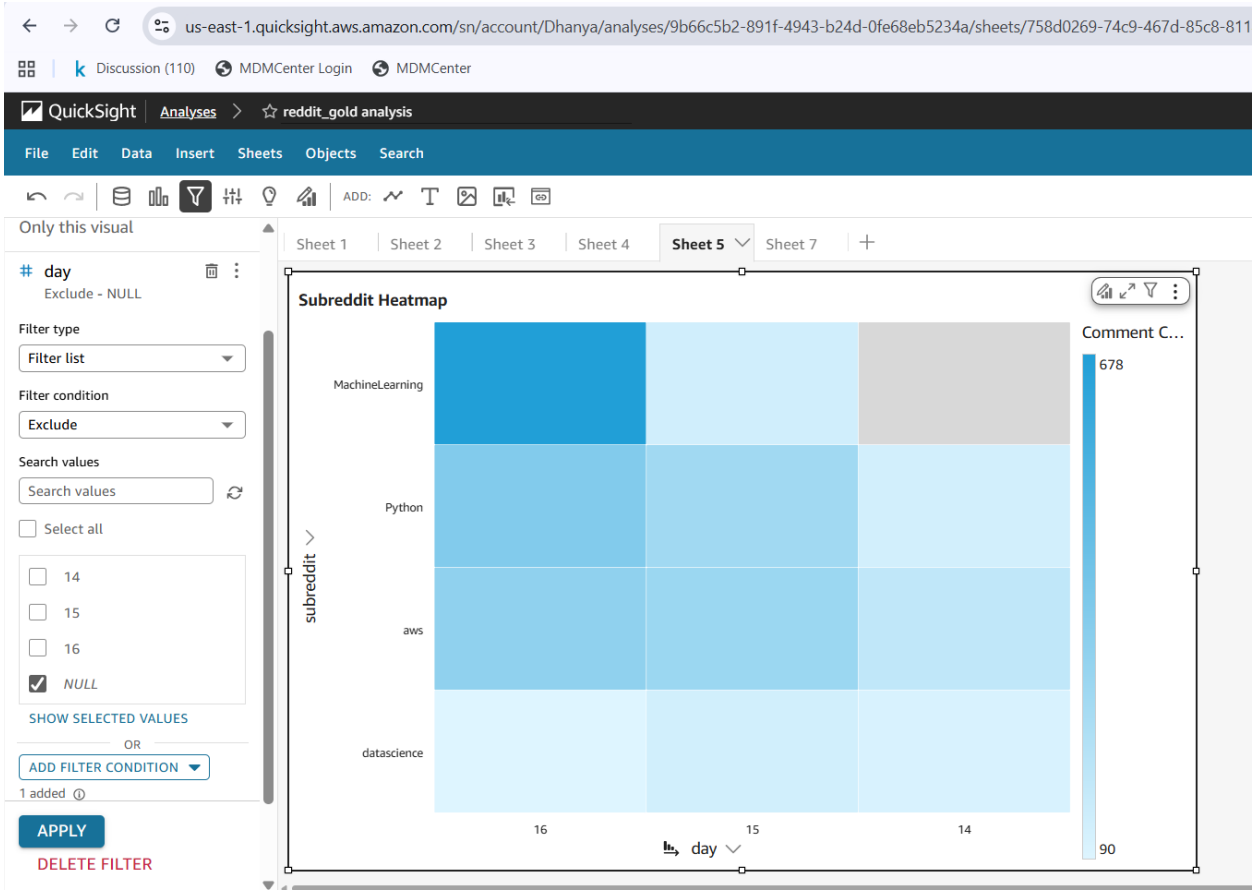
- Rows: subreddit
- Columns: day
- Values: comment_count (Sum)
- Color gradient: Activity intensity (light → dark blue)

Purpose:

The heatmap provides a time vs subreddit activity view, helping detect when each subreddit is most active.

Insights:

- MachineLearning shows strong peaks on certain days.
- Python and AWS communities are consistently active.
- Some missing values (null days) suggest days with no data collection or very low activity.
- Helps in identifying daily cycles of subreddit activity.



8.6 Scatter Plot (Sentiment vs Comments)

Description:

- X-axis: avg_sentiment (Sum)
- Y-axis: comment_count (Sum)
- Color: subreddit

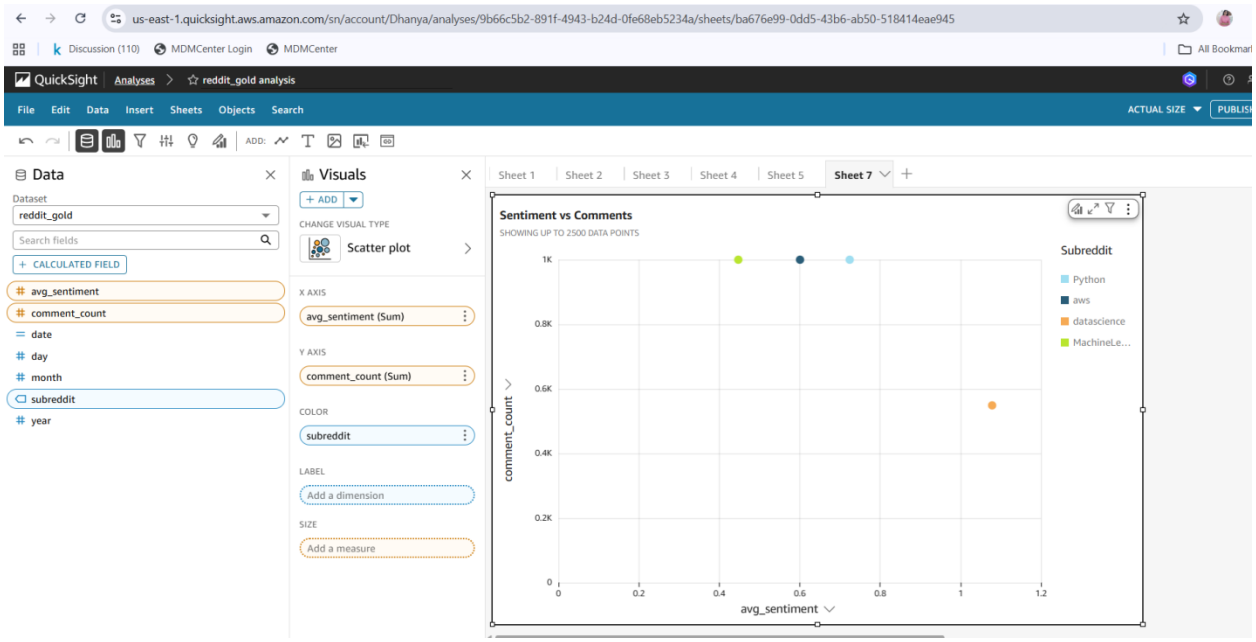
Purpose:

This scatter plot explores the relationship between sentiment and engagement.

Insights:

- High comment count + neutral sentiment indicates balanced discussions.
- DataScience appears with higher sentiment but fewer comments.
- AWS and Python have higher activity but lower sentiment scores.

- Useful for finding outliers (e.g., subreddits with high positivity but low activity).



9. Troubleshooting steps

Step/Component	Issue	Error/Message	Cause	Fix/Resolution
Reddit API (PRAW)	MissingRequiredAttributeException: client_id missing	PRAW could not find client_id	Env vars not loaded into script	Use .env with python-dotenv and pass values explicitly in DAG
Airflow DAG Import	ModuleNotFoundError: No module named 'utils'	DAG failed to import	Wrong mount path inside container	Mount project root (.:/opt/airflow) and place utils/ at root
Airflow Webserver Port	Bind for 0.0.0.0:8080 failed: port already allocated	Airflow container won't start	Port 8080 already in use	Change to 8081:8080 in docker-compose.yml

Step/Component	Issue	Error/Message	Cause	Fix/Resolution
LocalFilesystemToS3Operator	S3HookUriParseFailure: Please provide a bucket name using valid format	Wrong key value	Provided reddit/bronze/... without bucket	Always pass bucket separately and only the key (e.g., bronze/...)
GlueJobOperator	Invalid type for parameter Command.ScriptLocation	Airflow tried to create a new Glue job	Glue job already exists in console	Use boto3 start_job_run instead of GlueJobOperator for existing jobs
Glue IAM Role	Invalid type for parameter RoleName, value: None	Role not passed	Env var not picked in container	Hardcode role in DAG or load from airflow.env correctly
Glue Start Job Run	EntityNotFoundException: Failed to start job run due to missing metadata	Airflow failed to trigger job	Wrong job name or region mismatch	Double-check GLUE_SILVER_JOB, GLUE_GOLD_JOB values and AWS region
Athena Queries	Table not found	Athena couldn’t query	Partitions not refreshed	Run MSCK REPAIR TABLE before querying
QuickSight Date Field	parseDate syntax error	Calculated field failed	Used multi-line concat or wrong format string	Use parseDate(concat(toString({year}), '-', toString({month}), '-', toString({day})), 'yyyy-M-d')

Step/Component	Issue	Error/Message	Cause	Fix/Resolution
QuickSight Heatmap	Null column for day	Heatmap shows "null"	Missing partition values in data	Filter nulls or ensure Glue job writes all partitions

10. Cost Optimization Guide

Service	Potential Cost Driver	Optimization Strategy
Amazon S3 (Bronze/Silver/Gold layers)	Storing large raw/processed files indefinitely	- Enable S3 Lifecycle policies (move Bronze → Glacier after 30 days)- Store Silver/Gold in Parquet instead of CSV (columnar, compressed)
AWS Glue ETL Jobs	Long job runtimes and large DPUs	- Use smaller DPU allocation (2 instead of 10) for light jobs- Enable Job Bookmarks to process only new data- Reuse scripts via script editor instead of auto-generated
AWS Glue Crawlers	Frequent unnecessary crawls	- Schedule crawlers only after new partitions arrive (daily, not hourly)- Use MSCK REPAIR TABLE instead of crawler if schema stable
Amazon Athena	Large query scans due to unpartitioned data	- Partition tables by year, month, day- Use Parquet + Snappy compression- Use SELECT columns instead of SELECT *
Amazon QuickSight	Reader session costs ($0.30/session)	- Use SPICE datasets (in-memory, cheaper for repeated queries)- Share dashboards with limited reader accounts only
Airflow (Docker Compose local)	Running containers continuously	- Stop containers when not in use (docker-compose down)- For production: deploy on

Service	Potential Cost Driver	Optimization Strategy
		Fargate with auto-scaling
IAM/Permissions	Overprovisioned roles may lead to indirect costs (accidental usage)	- Use least-privilege IAM (Glue can only access required S3 buckets)- Rotate keys, disable unused creds
Logging & Monitoring (CloudWatch)	Excessive logs from Glue/Athena	- Enable log retention policy (keep 30 days)- Export important logs to S3 Glacier instead of storing indefinitely

11. Git Repo

The complete code is available in git. The repository url is:

https://github.com/Dhanyamol-Devassy/reddit-comments-pipeline-aws/

12. Conclusion

This project successfully demonstrated an end-to-end data engineering pipeline on AWS using Reddit comments as the data source. Starting from raw ingestion through the Reddit API, the pipeline applied modern data engineering practices — Bronze/Silver/Gold architecture, partitioned storage in S3, Glue-based ETL, Athena queries, and QuickSight visualizations.

Key outcomes:

- Built a fully automated Airflow DAG that orchestrates all stages (extract → upload → transform → validate).
- Designed Glue ETL jobs that flatten, clean, and enrich Reddit data efficiently, while leveraging partitioning and columnar formats (Parquet) to optimize performance and cost.
- Created Athena tables that enabled SQL-style queries directly on S3, minimizing the need for expensive database infrastructure.
- Delivered actionable QuickSight dashboards, offering multiple perspectives — comment volumes, sentiment distribution, temporal trends, and subreddit-level comparisons.

- Documented detailed troubleshooting steps and cost optimization strategies, ensuring the pipeline is both reliable and production-ready.

Ultimately, this pipeline is extensible — more subreddits or timeframes can be added easily, and the Gold dataset can support advanced analytics such as topic modeling, anomaly detection, or forecasting. This foundation proves the viability of building scalable, cost-effective, and insightful cloud-based data platforms.