

Data Engineering YouTube Analysis Project

Harshada Rawool

Problem statement:


The increasing volume of YouTube video content presents challenges in data management, processing, and analysis. Current systems struggle to efficiently handle and analyze large datasets, leading to missed opportunities for gaining insights into video performance and trends. Specifically, there is a need to:

- Securely manage vast amounts of YouTube video data to ensure data integrity and privacy.
- Streamline the processing of both structured and semi-structured data to reduce latency and improve efficiency.
- Analyze video categories and trending metrics to identify key factors driving video popularity and viewer engagement.

This project aims to develop a robust data engineering solution that addresses these challenges, providing a comprehensive framework for managing, processing, and analyzing YouTube video data.



Project goals:

1. **Data Ingestion:** Develop a robust pipeline to ingest data seamlessly from various sources, ensuring data consistency and quality.
 2. **ETL System:** Implement an efficient ETL (Extract, Transform, Load) system to convert raw data into a structured, usable format.
 3. **Centralized Data Lake:** Establish a centralized data lake to store and manage data from multiple sources, facilitating easy access and analysis.
 4. **Scalability:** Design the system to scale seamlessly with growing data volumes, ensuring consistent performance and reliability.
 5. **Cloud Integration:** Leverage AWS cloud services to handle large-scale data processing, ensuring scalability, flexibility, and cost-efficiency.
 6. **Advanced Analytics and Reporting:** Develop an interactive dashboard to visualize data insights and trends, enabling informed decision-making and strategic planning.
- 

Services We Will Be Using

Amazon S3: An object storage service offering industry-leading scalability, data availability, security, and performance, essential for storing and managing vast amounts of data.

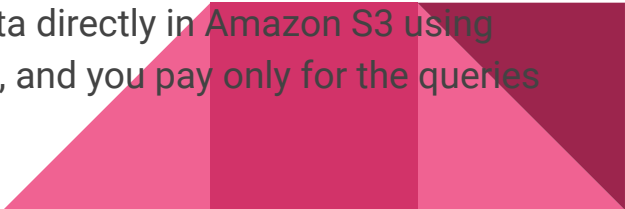
AWS IAM: Identity and Access Management service that enables secure management of access to AWS services and resources, ensuring only authorized users and applications can access sensitive data.

Amazon QuickSight: A scalable, serverless, machine learning-powered business intelligence (BI) service built for the cloud, designed to create and share rich visualizations and insights.

AWS Glue: A serverless data integration service that simplifies discovering, preparing, and combining data for analytics, machine learning, and application development, streamlining the ETL process.

AWS Lambda: A compute service that allows you to run code without provisioning or managing servers, enabling event-driven data processing and scaling automatically to handle varying workloads.

AWS Athena: An interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL. Athena is serverless, so there's no infrastructure to manage, and you pay only for the queries you run.



Dataset Used

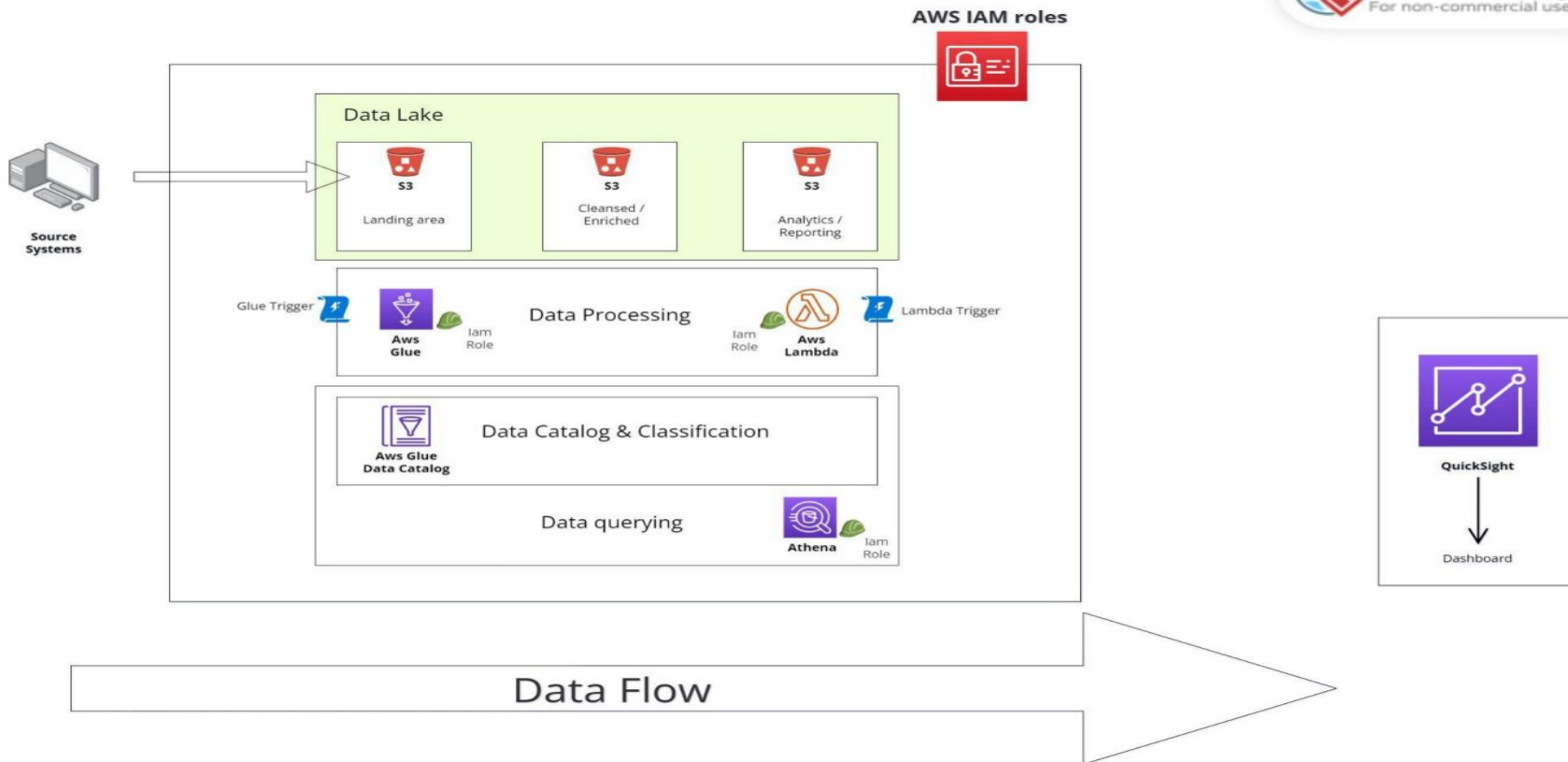
This Kaggle dataset contains statistics (CSV files) on daily popular YouTube videos over the course of many months. There are up to 200 trending videos published every day for many locations. The data for each region is in its own file. The video title, channel title, publication time, tags, views, likes and dislikes, description, and comment count are among the items included in the data. A category_id field, which differs by area, is also included in the JSON file linked to the region.

Data is collected using youtube's API

<https://www.kaggle.com/datasnaek/youtube-new>



Data Flow



Create S3 bucket as a landing data storage

I am using aws cli to copy raw data into s3 bucket.

(raw data bucket=json and csv)


```
# To copy all JSON Reference data to same location:  
aws s3 cp . s3://de-on-youtube-raw-useast1-dev/youtube/raw_statistics_reference_data/ --recursive --exclude "*" --include "*.json"
```

```
# To copy all data files to its own location, following Hive-style patterns:  
aws s3 cp CAVideos.csv s3://de-on-youtube-raw-useast1-dev/youtube/raw_statistics/region=ca/  
aws s3 cp DEVideos.csv s3://de-on-youtube-raw-useast1-dev/youtube/raw_statistics/region=de/  
aws s3 cp FRVideos.csv s3://de-on-youtube-raw-useast1-dev/youtube/raw_statistics/region=fr/  
aws s3 cp GBVideos.csv s3://de-on-youtube-raw-useast1-dev/youtube/raw_statistics/region=gb/
```

All JSON data is uploaded in the same directory, while CSV data will be placed in their respective region directories. This arrangement ensures that when the data is imported into the Glue data catalog, it will maintain partitioning based on region.

Cleaning procedure:

Our cleaning procedure includes 2 steps.

- First we'll clean the **json data** containing categories and store them in one table.
 - Secondly, we'll clean the **csv data** which contains the videos information and store them in another table.
 - Later we'll inner join both the tables using **category_id** and id as references.
 - Finally, this table is ready to visualize.
- 

Build glue crawler and catalog

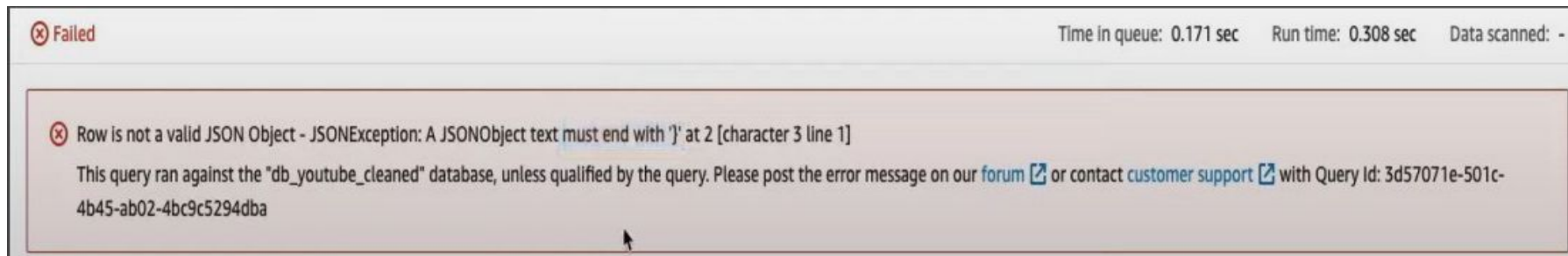
- Initially a crawler(raw-json) will parse the raw json data and create a catalog around it making it ready for further operations.
- An IAM role for Glue with permissions for S3FullAcces is set up to keep resource access restricted.
- The crawler will parse the data and will save it under a Table in Glue Database



Querying Data in AWS Athena from Glue Catalog

We are querying the JSON data cataloged by AWS Glue using AWS Athena.

Initially, if we run a query to show table data , we'll encounter an error. The reason behind this is the data we require is contained in an array.



AWS Glue may not be able to understand the desired data without pre-processing

```
{
  "kind": "youtube#videoCategoryListResponse",
  "etag": "\"1d9biNPKjAjjV7EZ4EKeEGrhao/1v2mrzYSYG6onNlt2qTj13hkQZk\"",
  "items": [
    {
      "kind": "youtube#videoCategory",
      "etag": "\"1d9biNPKjAjjV7EZ4EKeEGrhao/Xy1mB4_yLrHy_BmKmPBggty2mZQ\"",
      "id": "1",
      "snippet": {
        "channelId": "UCBR8-60-B28hp2BmDPdntcQ",
        "title": "Film & Animation",
        "assignable": true
      }
    },
    {
      "kind": "youtube#videoCategory",
      "etag": "\"1d9biNPKjAjjV7EZ4EKeEGrhao/UZ1oLIi2zdxIh045ZTFR3a3NyTA\"",
      "id": "2",
      "snippet": {
        "channelId": "UCBR8-60-B28hp2BmDPdntcQ",
        "title": "Autos & Vehicles",
        "assignable": true
      }
    }
  ]
}
```

We only require the data in green but glue fails to do so.

The reason behind why Glue is not able to parse this data is because of a thing called Json SerDe (serialize and deserialize).

- In layman terms , glue requires all its contents in one single line rather than in multiple lines.

The following example will work.

```
{ "key" : 10 }
{ "key" : 20 }
```

The following example will not work.

```
{
  "key" : 10
}
{
  "key" : 20
}
```

To fix this.....

Converting json data into apache parquet format:

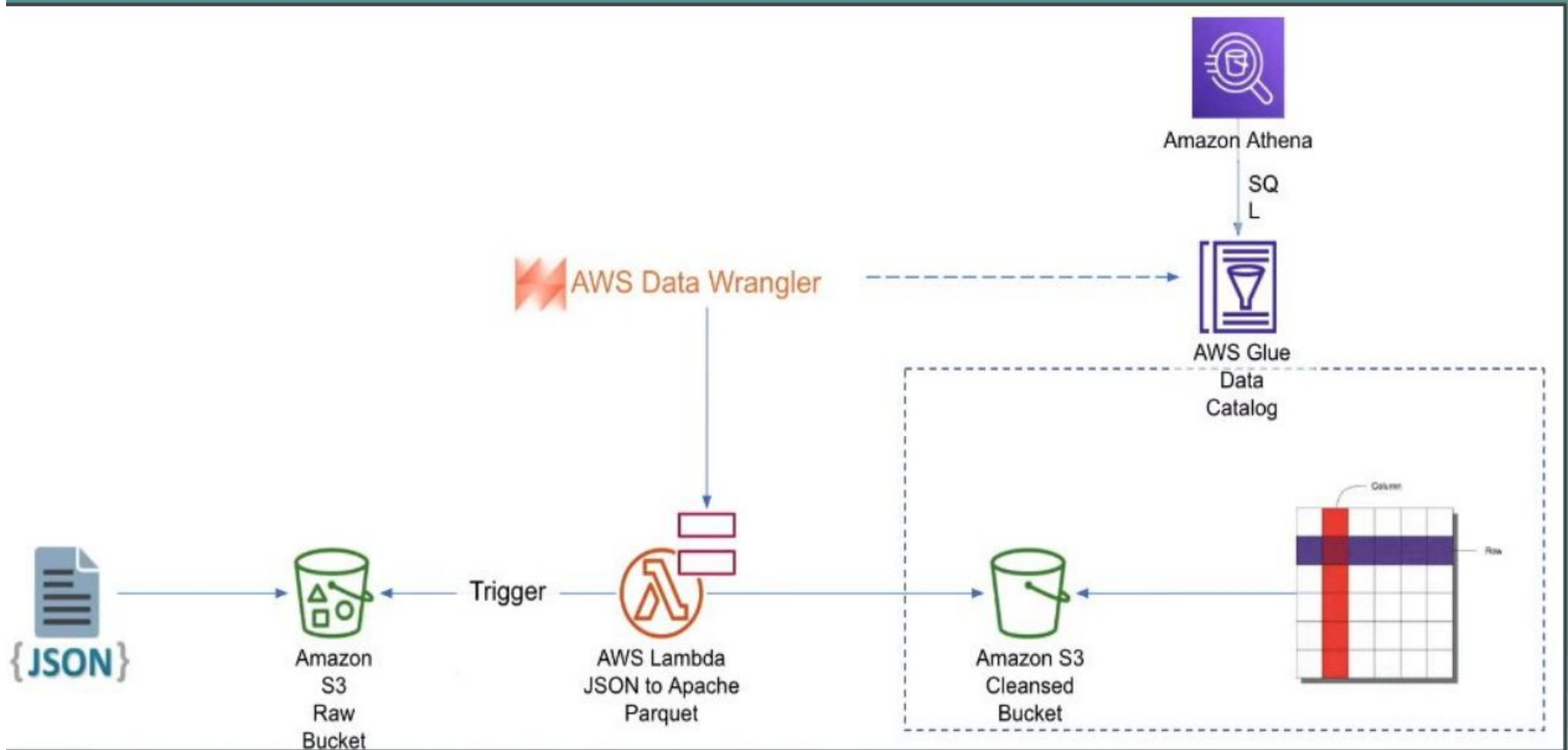
Pre-processing or data cleaning is required to extract specific data from a JSON file.(Because of Serde:serialise and deserialise).

```
C:\> Users > HARSHADA > AppData > Local > Temp > c94db840-48ae-4da8-b3e1-342cf02dc208_archive.zip.208 > CA_category_id.json >
1 {
2   "kind": "youtube#videoCategoryListResponse",
3   "etag": "\"ld9biNPKjAjjV7EZ4EKeEGrhao/1v2mrzYSYG6onMLT2qTj13hkQZk\"",
4   "items": [
5     {
6       "kind": "youtube#videoCategory",
7       "etag": "\"ld9biNPKjAjjV7EZ4EKeEGrhao/Xy1mB4_yLrHy_BmKmpBggtY2mZQ\"",
8       "id": "1",
9       "snippet": {
10        "channelId": "UCBR8-60-B28hp2BmDPdntcQ",
11        "title": "Film & Animation",
12        "assignable": true
13      },
14    },
15    {
16      "kind": "youtube#videoCategory",
17      "etag": "\"ld9biNPKjAjjV7EZ4EKeEGrhao/UZ1oLIi2dxIh045ZTFR3a3NyTA\"",
18      "id": "2",
19      "snippet": {
20        "channelId": "UCBR8-60-B28hp2BmDPdntcQ",
21        "title": "Autos & Vehicles",
22        "assignable": true
23      },
24    },
25    {
26      "kind": "youtube#videoCategory",
27      "etag": "\"ld9biNPKjAjjV7EZ4EKeEGrhao/nqRIq97-xe5XRZTxbkNKFVe5Lmg\"",
28      "id": "10",
29      "snippet": {
30        "channelId": "UCBR8-60-B28hp2BmDPdntcQ",
31        "title": "Music",
32        "assignable": true
33      },
34    }
35  ]
36 }
```

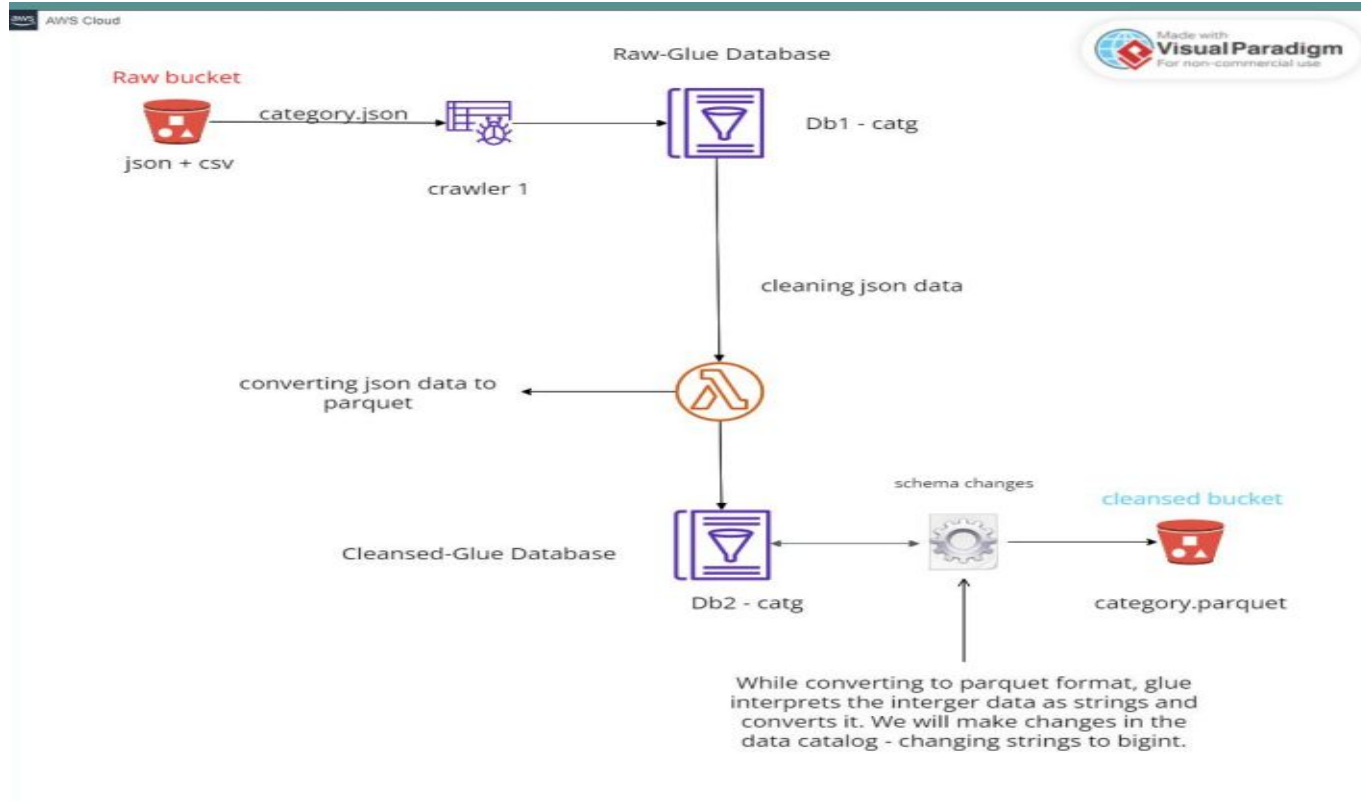


a	b	c	...	zf
a1	b1	c1	...	zf1
a2	b2	c2	...	zf2
a3	b3	c3	...	zf3
a4	b4	c4	...	zf4
a5	b5	c5	...	zf5
a6	b6	c6	...	zf6
a7	b7	c7	...	zf7

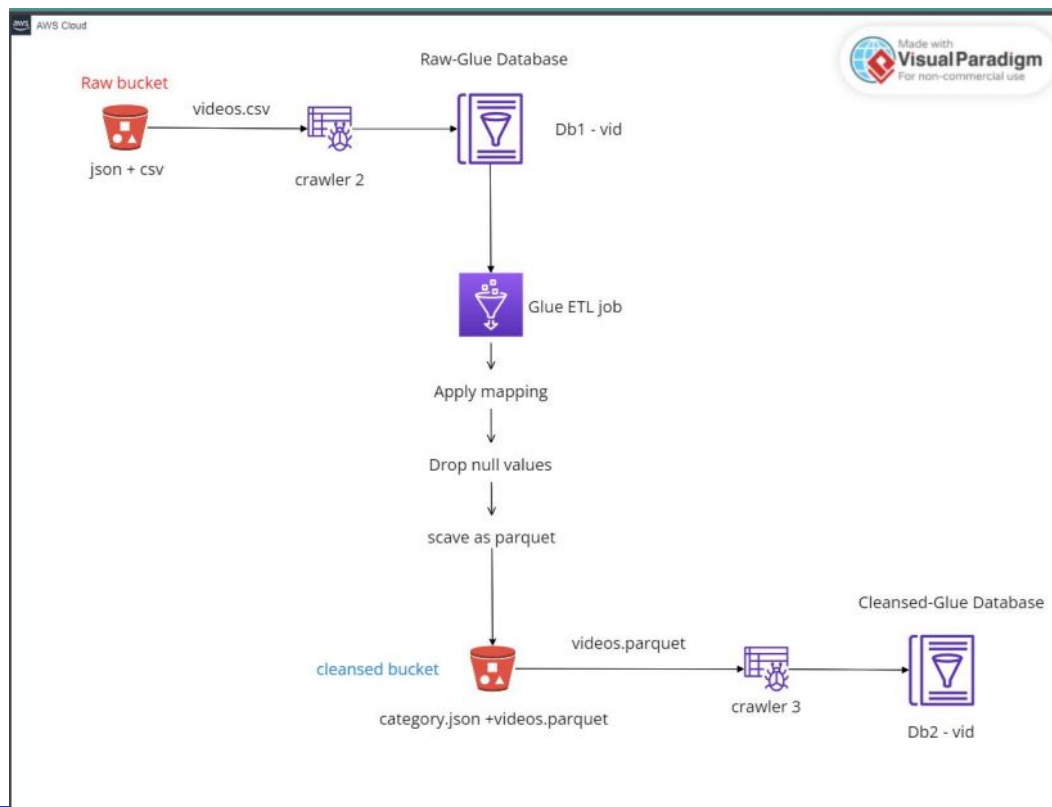
Data cleaning : Semi-structured data to Structured data pipeline



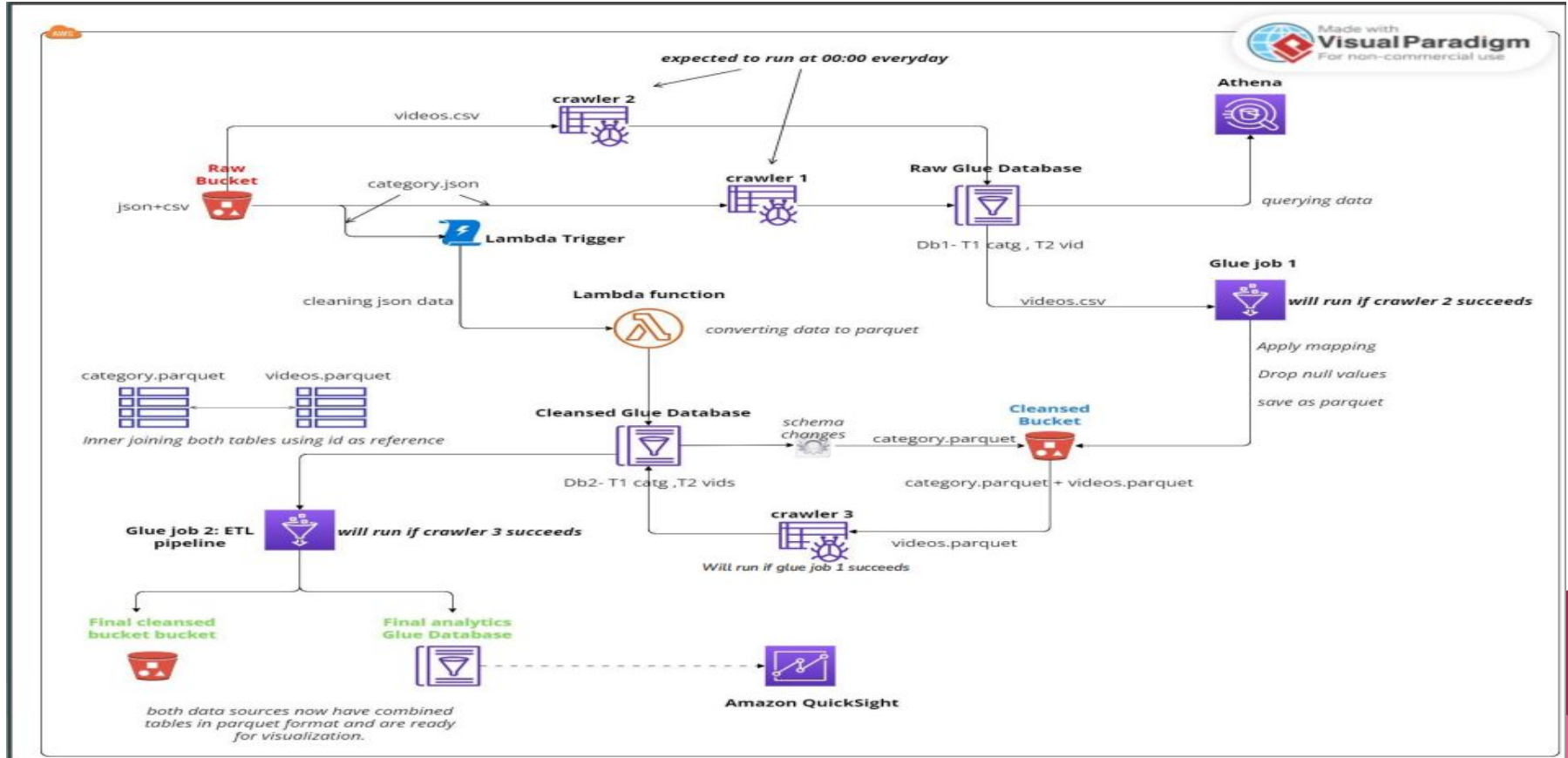
Data cleansing 1 : cleaning json data



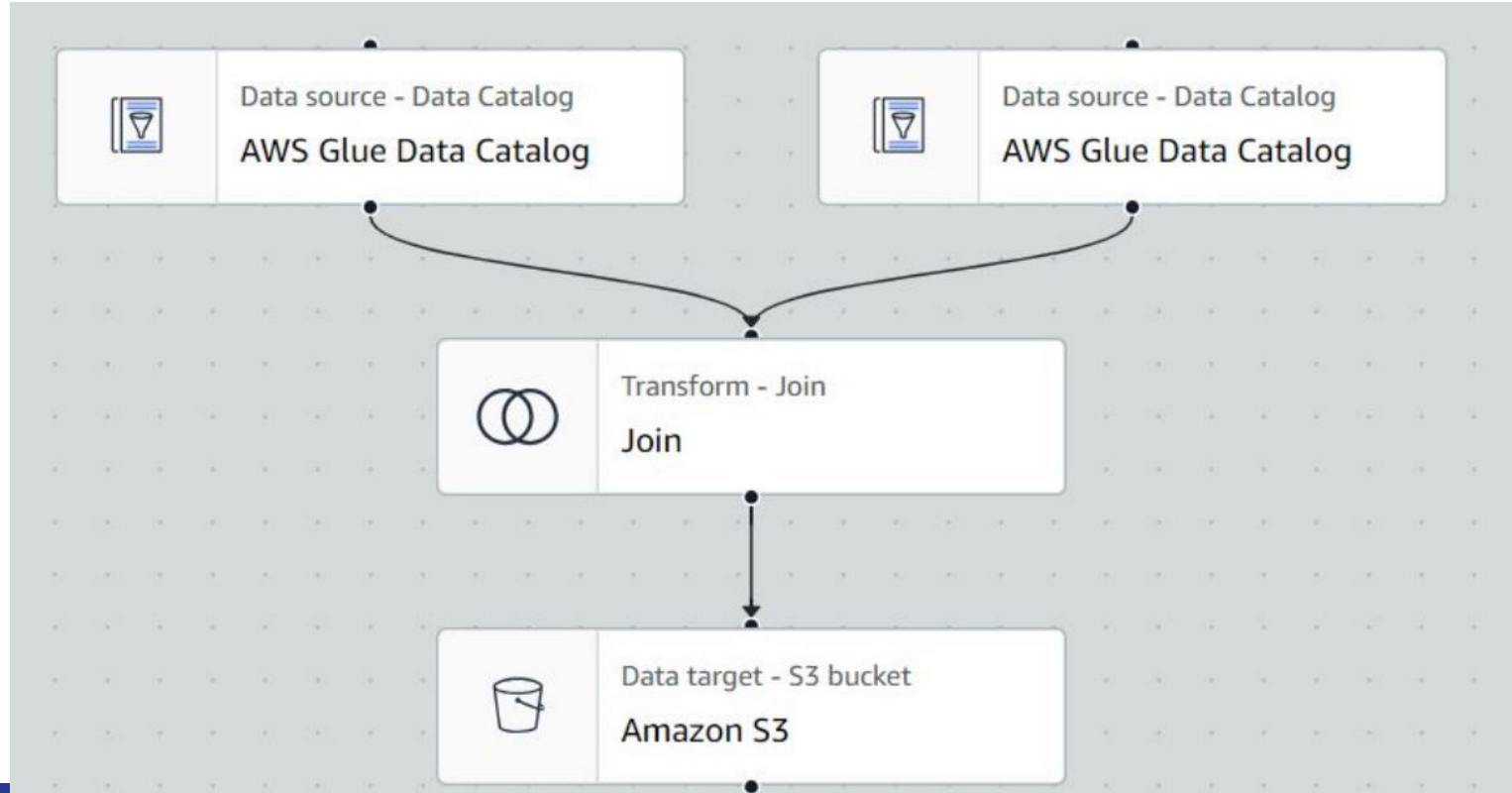
Data cleansing 2: cleaning csv data



Data flow



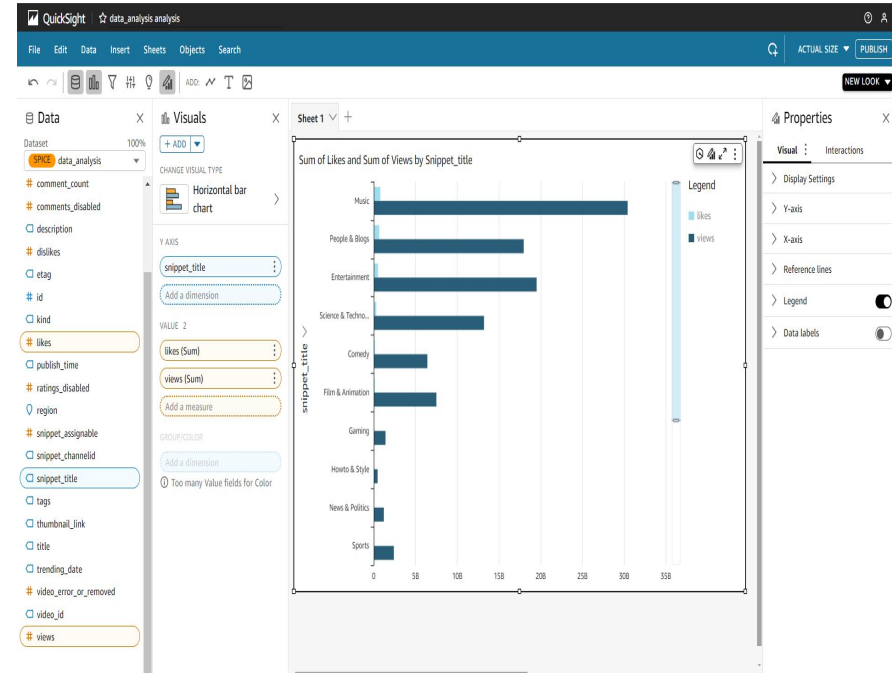
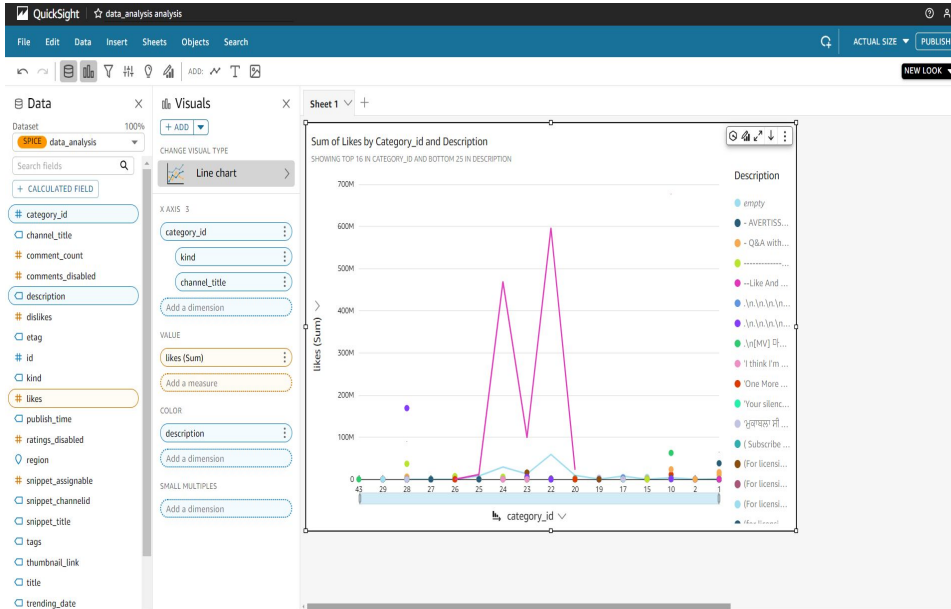
Glue : ETL pipeline for generating final parquet cleansed data



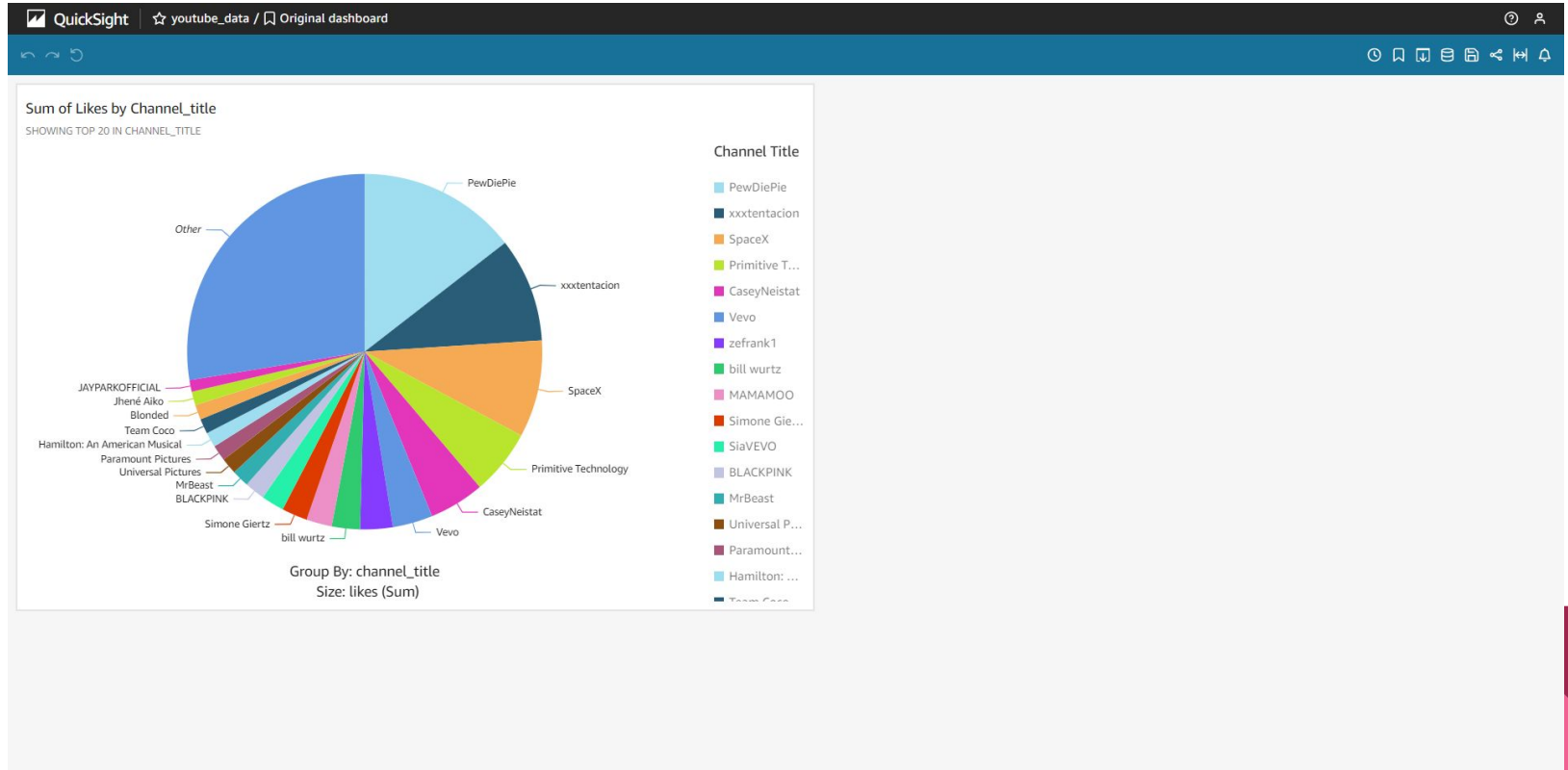
Lets see on Quick site



Analysis



Dashboard



THANK YOU

