Youtube Trending Data Analysis

Aws project

- Aryan Kamble

Problem statement

The phenomenon of YouTube trending videos varies significantly across different regions, influenced by cultural preferences, language, and regional trends. However, there exists a gap in understanding how various factors such as likes, comments, video categories, and language contribute to the trending status of YouTube videos within specific regions.

This project aims to address this gap by conducting a detailed analysis of YouTube trending data, focusing on regional nuances and the influence of key factors.

Objectives

- **Regional Variation**: Explore how trending videos differ across regions in categories, language, and engagement metrics.
- **Likes and Comments**: Assess the impact of likes and comments on trending likelihood in different regions.
- **Category Preferences**: Identify prevalent video categories and their regional popularity.
- **Language Influence**: Analyze how video language affects trending in specific regions.
- **Cross-Regional Trends:** Investigate global trends shaping regional YouTube dynamics.

Project Solutions and Goals

Data Ingestion
Ingest data,one-offs and incrementally

ETL Design
Extract, transform and load data efficiently

Data Lake
Design and build a new Data Lake architecture

Scalability
The data architecture should scale efficiently

AWS cloud
AWS as the cloud provider

Reporting
Build a Business Intelligence tier , incl Dashboards

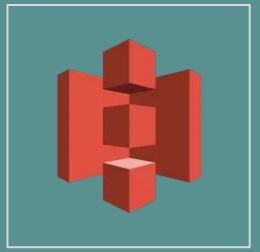
Youtube Dataset

In this project , I’m using the Youtube Trending videos dataset from Kaggle.

This dataset contains 2 types of files partitioned based on regions - category.json and videos.csv. These files contain data like video category ,number of views, shares , comments ,likes ,etc.

Data is collected by Youtube’s own API.

- AWS S3 as an ingestion point for our raw data



  - Data will be uploaded here using AWS CLI
  - Later, the cleansed analytical data will also be stored here in separate bucket.

```
# To copy all JSON Reference data to same location:
aws s3 cp . s3://de-on-youtube-raw-useast-1-aryan/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-useast-1-aryan/youtube/raw_statistics/region=ca/
aws s3 cp DEvideos.csv s3://de-on-youtube-raw-useast-1-aryan/youtube/raw_statistics/region=de/
aws s3 cp FRvideos.csv s3://de-on-youtube-raw-useast-1-aryan/youtube/raw_statistics/region=fr/
```

Uploading data to S3 using the aws s3 cp command.

Raw-data-bucket (R-1(json + csv)).

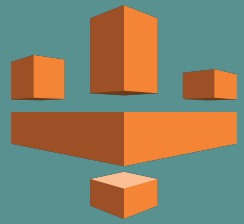
Note : All json data is uploaded in the same directory , wheres csv data will be placed in their region directory respectively.

I'm uploading this data in such a way that later when my data is imported into the Glue data catalog , it will maintain its 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.

- AWS Glue Data Catalog for data warehousing.
    - 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 (D1-T1(json))



AWS Glue

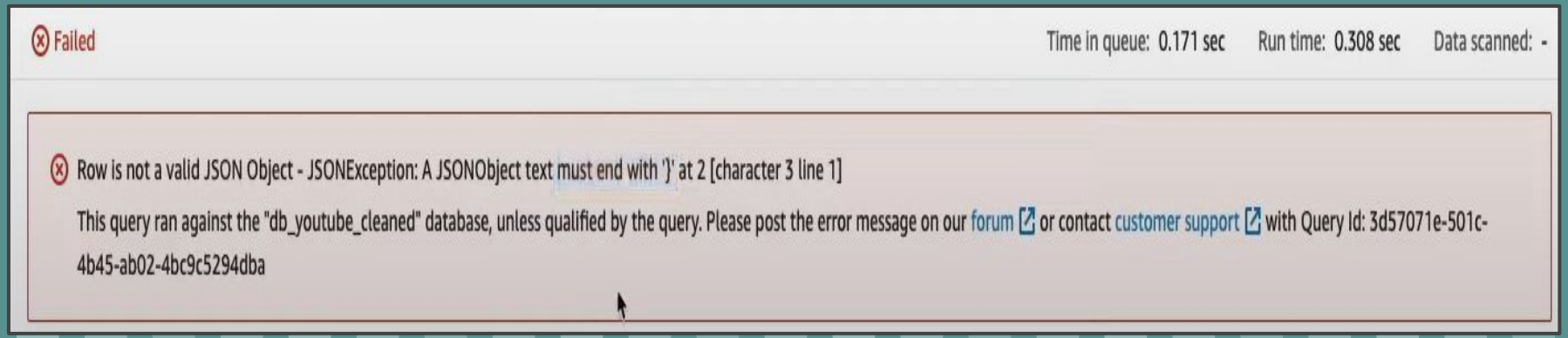
- AWS Athena for querying data .

    - We'll use Athena for querying our data . We'll have to create a new s3 bucket as it stores its query results. (Ath-1).

    - Tables from Glue database will be viewed and queried here.

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



```
{
 "kind": "youtube#videoCategoryListResponse",
 "etag": "\"ld9biNPKjAjgjV7EZ4EKeEGrhao/1v2mrzYSYG6onNLt2qTj13hkQZk\"",
 "items": [
  {
   "kind": "youtube#videoCategory",
   "etag": "\"ld9biNPKjAjgjV7EZ4EKeEGrhao/Xy1mB4_yLrHy_BmKmPBggty2mZQ\"",
   "id": "1",
   "snippet": {
    "channelId": "UCBR8-60-B28hp2BmDPdntcQ",
    "title": "Film & Animation",
    "assignable": true
   }
  },
  {
   "kind": "youtube#videoCategory",
   "etag": "\"ld9biNPKjAjgjV7EZ4EKeEGrhao/UZ1oLIIz2dxIhO45ZTFR3a3NyTA\"",
   "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.

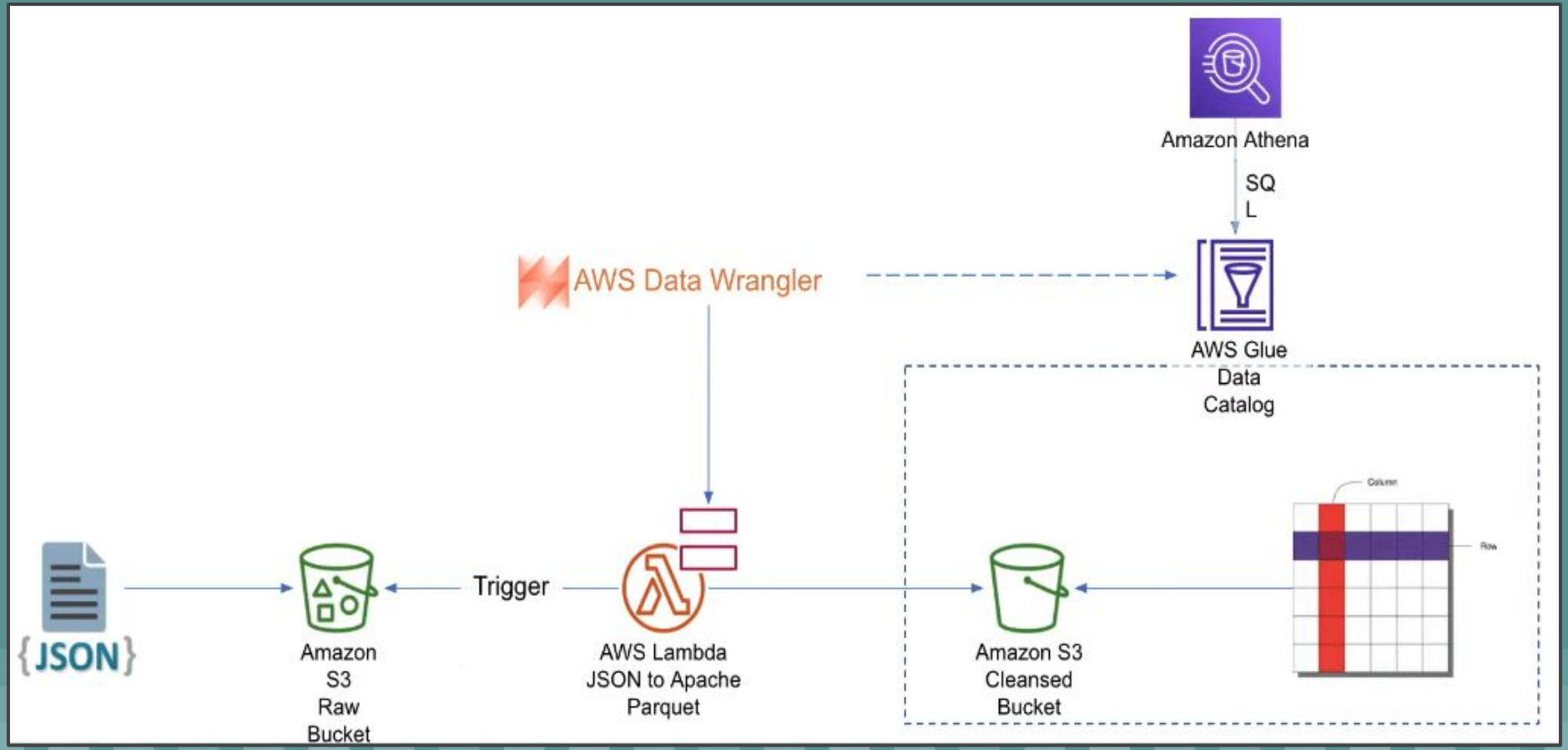
```
{
  "key" : 10
}
{
  "key" : 20
}
```

To fix this.....

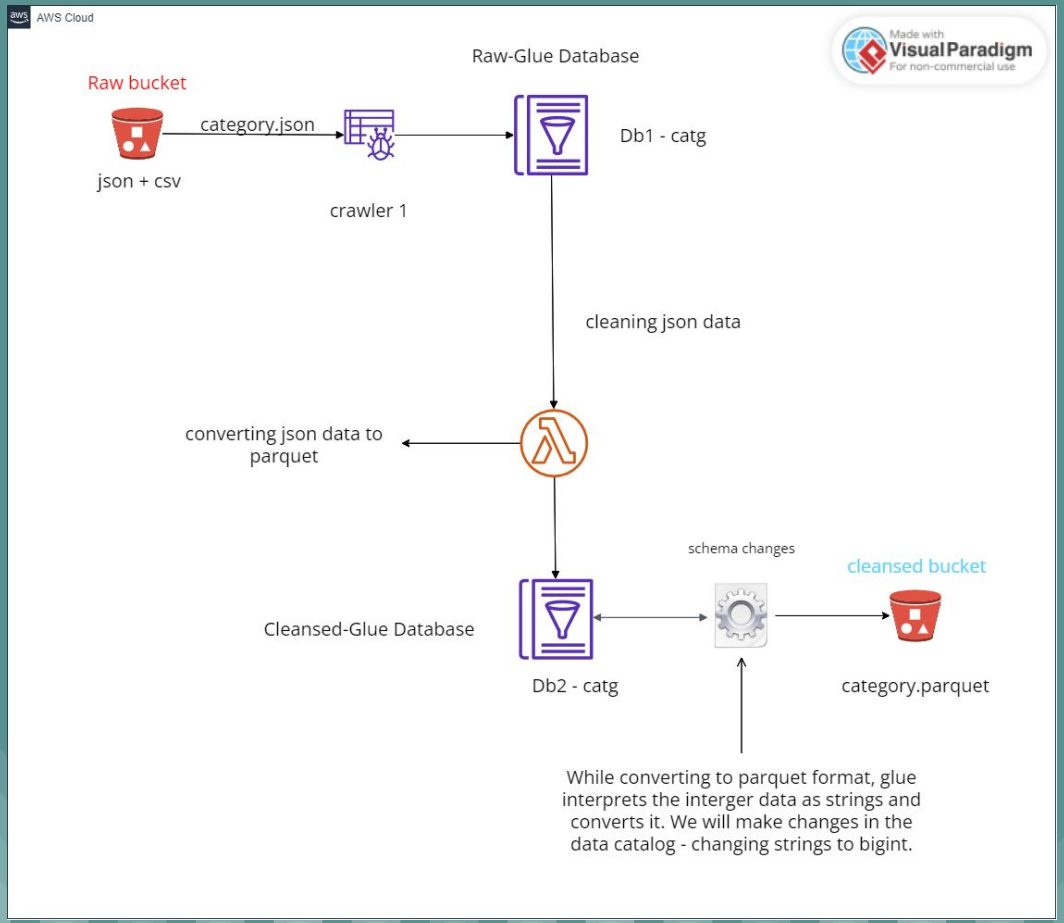
We will convert our JSON data to Apache Parquet format (tabular)

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

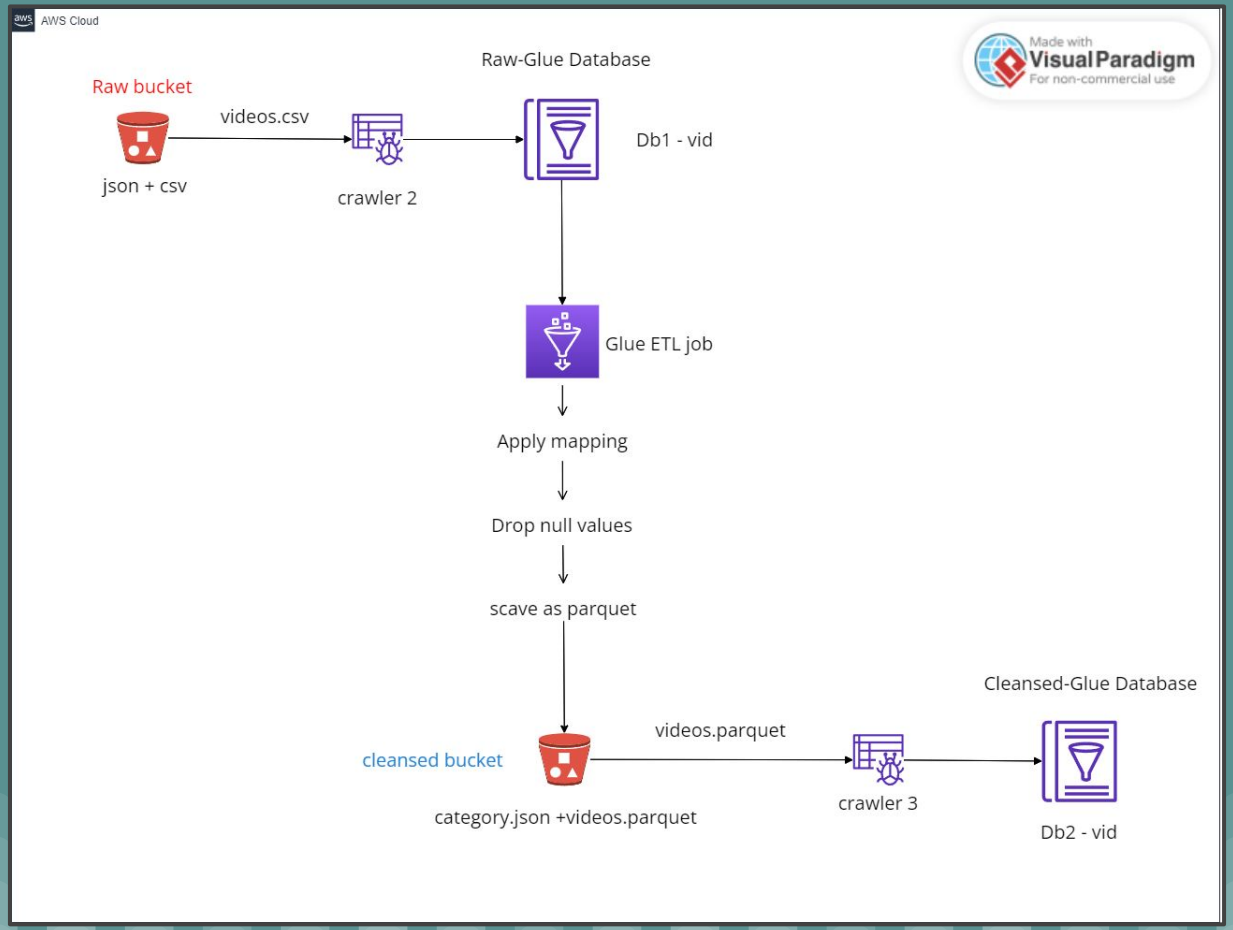
Data cleaning : Semi-structured data to Structured data pipeline



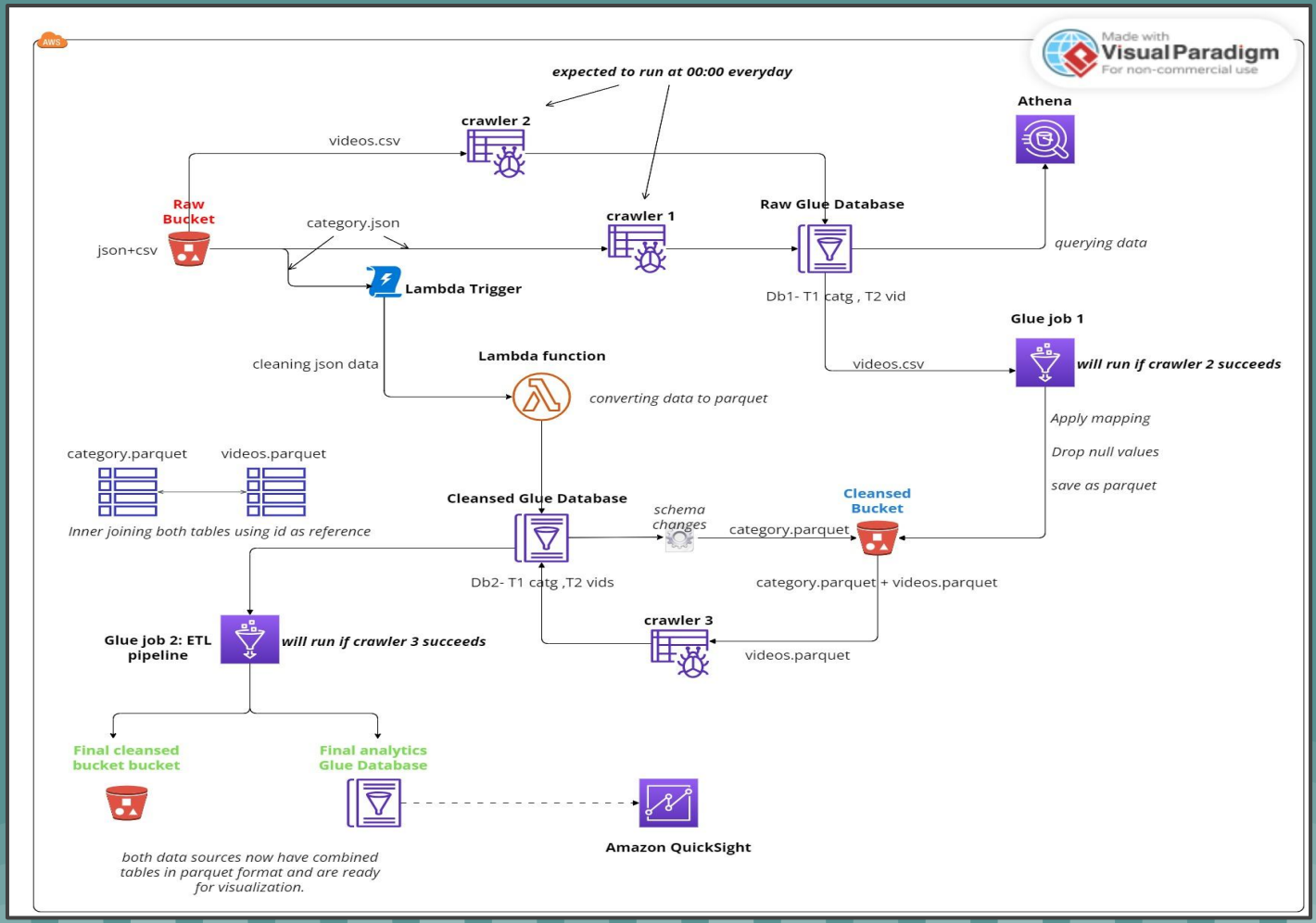
Data cleansing : cleaning json data



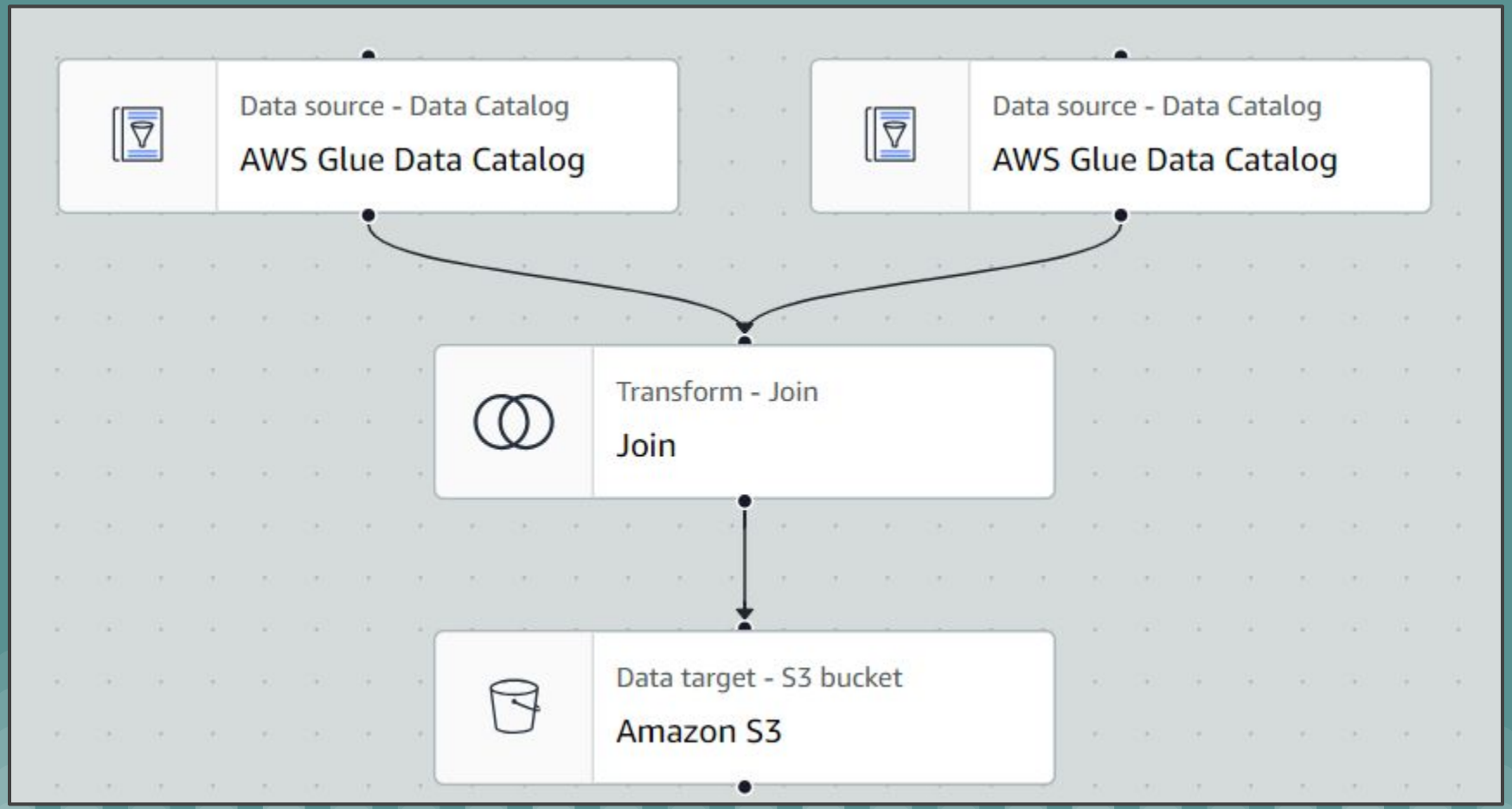
Data cleansing : cleaning csv data



Data flow

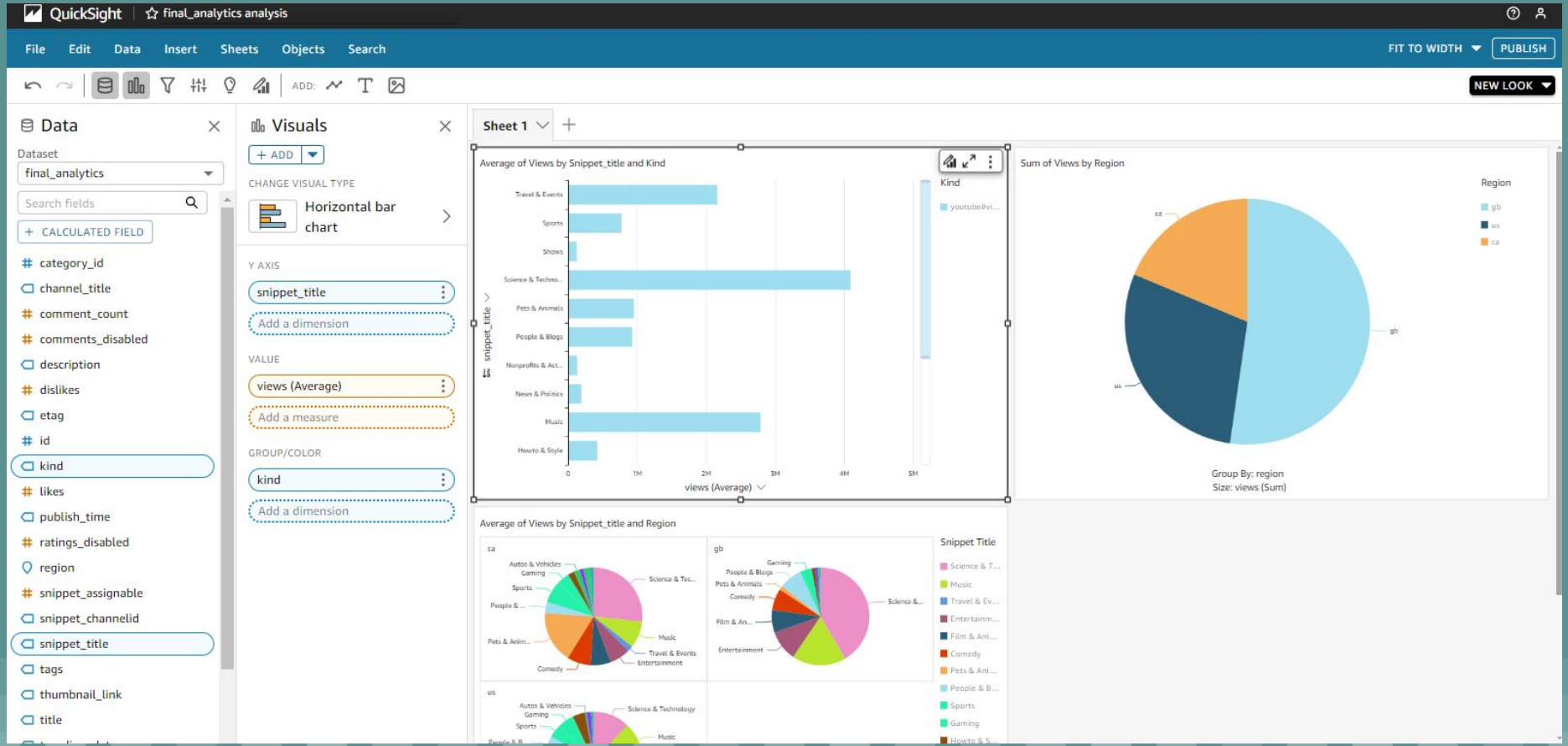

Made with Visual Paradigm For non-commercial use
expected to run at 00:00 everyday
crawler 2
videos.csv
Athena
Raw Bucket
json+csv
category.json
crawler 1
Raw Glue Database
querying data
Lambda Trigger
Db1- T1 catg , T2 vid
Glue job 1
cleaning json data
Lambda function
videos.csv
will run if crawler 2 succeeds
converting data to parquet
Apply mapping
Drop null values
save as parquet
category.parquet
videos.parquet
Cleansed Glue Database
schema changes
Cleansed Bucket
Inner joining both tables using id as reference
category.parquet
Db2- T1 catg ,T2 vids
category.parquet + videos.parquet
crawler 3
Glue job 2: ETL pipeline
will run if crawler 3 succeeds
videos.parquet
Final cleansed bucket bucket
Final analytics Glue Database
Amazon QuickSight
both data sources now have combined tables in parquet format and are ready for visualization.

Glue : ETL pipeline for generating final parquet cleansed data

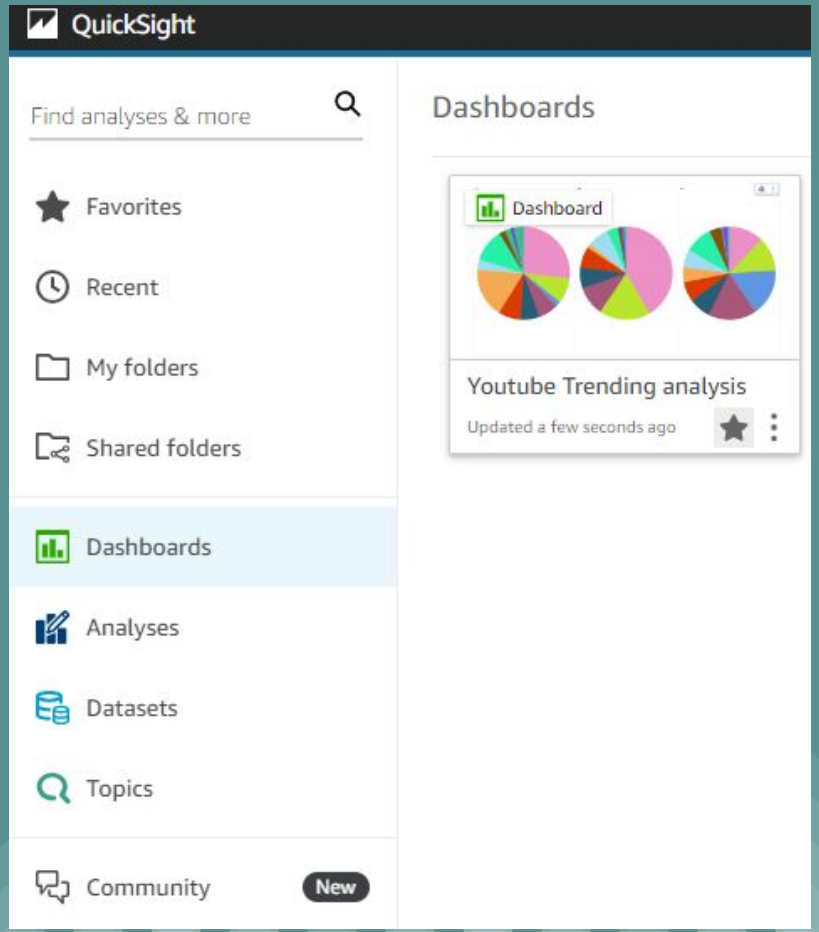


Moving on to AWS QuickSight

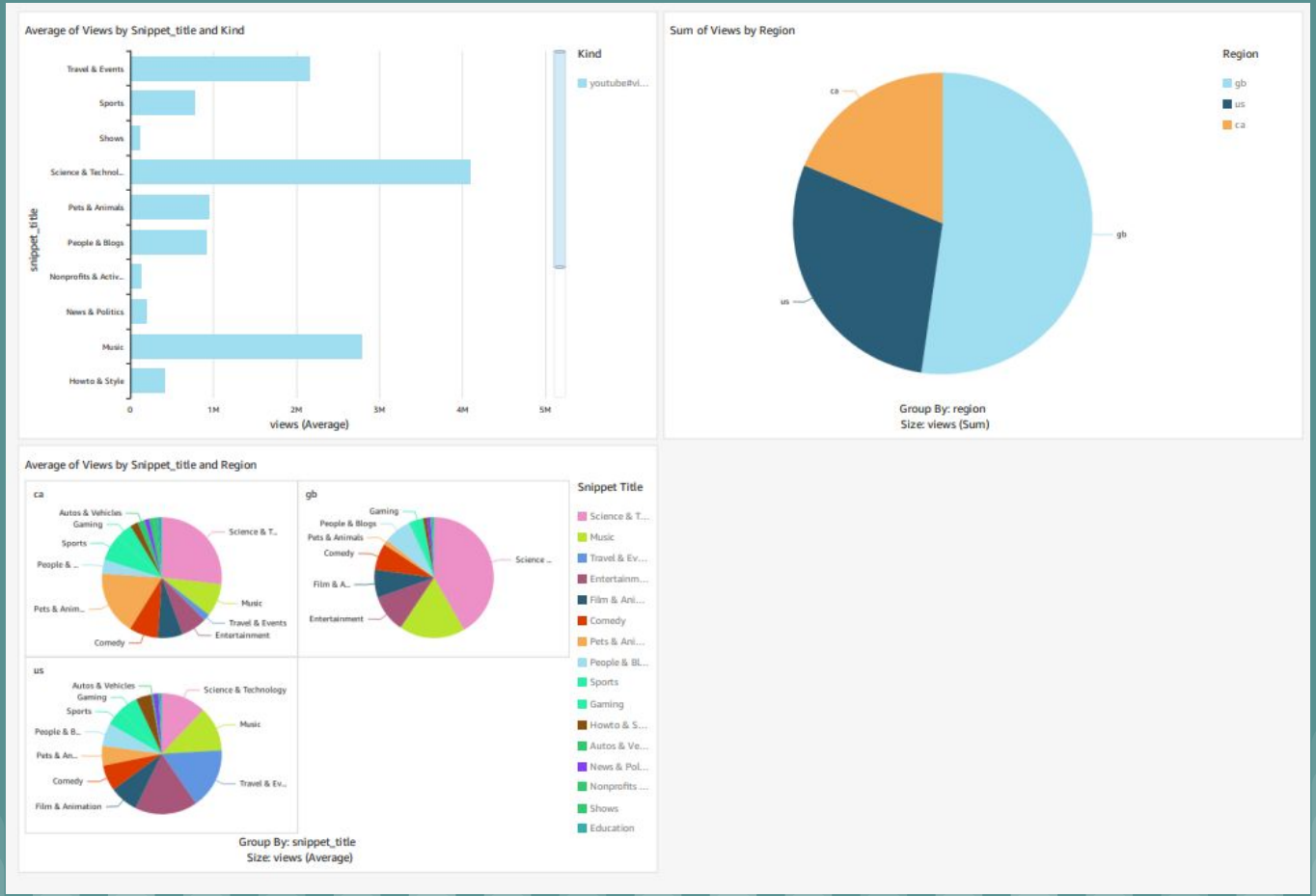
Quicksight : Analysis



Quicksight : Published Dashboard



Quicksight : charts



Thank you