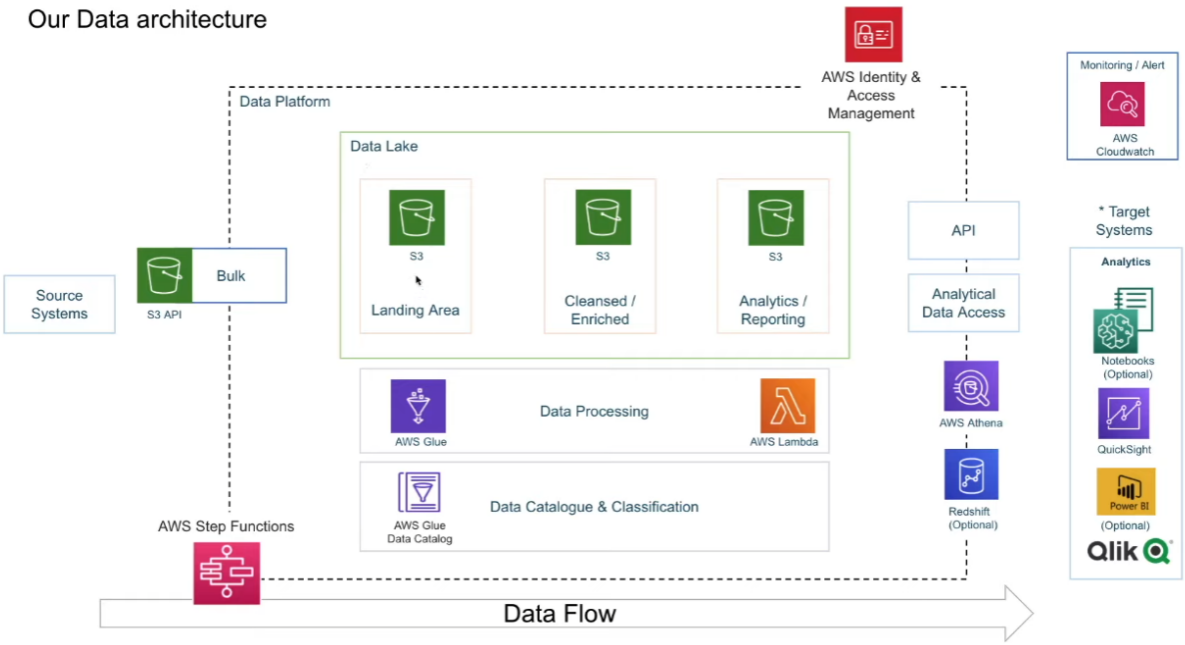
Youtube Data Analysis

1. Project Architecture.



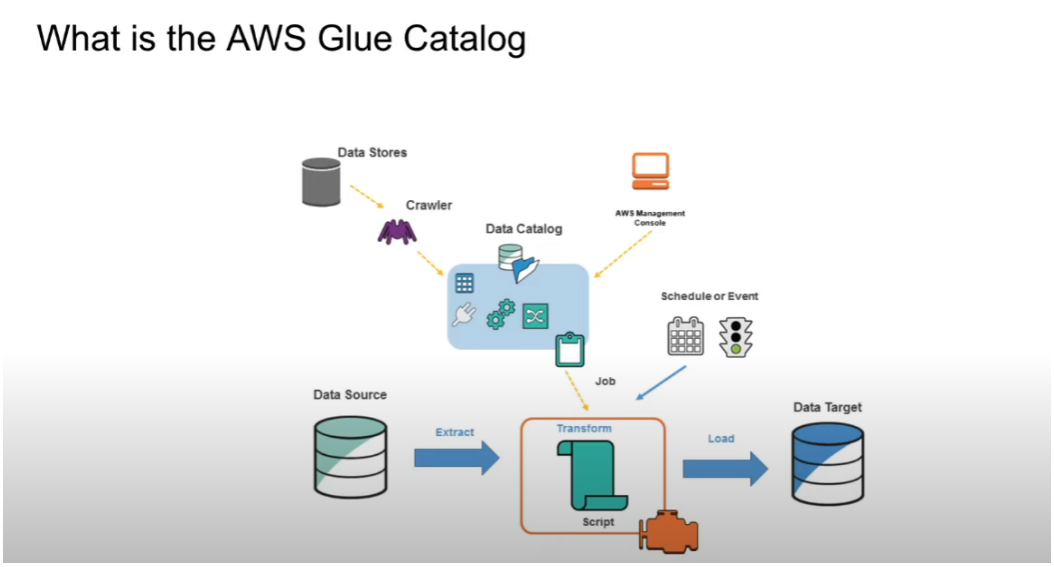
2. Information about Dataset.

a. Top Trending Videos.
b. What is Trending?
    i. YouTube uses factors, including users interactions e.g. number of views, shares, comments and likes.
    ii. Not the most-viewed videos overall for the calendar year
c. Source: Kaggle; Data Collected using Youtube API https://www.kaggle.com/datasnaek/youtube-new
d. Json File has each category attached in Data and CSV file has multiple video metadata partitioned by region.
e. The data also includes a `category_id` field, which varies between regions. To retrieve the categories for a specific video, find it in the associated `JSON`.

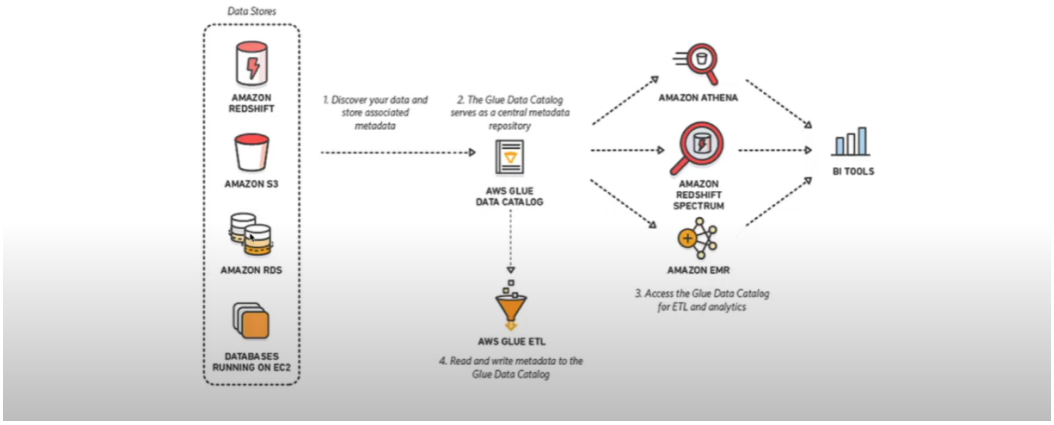
3. Interaction with AWS
    a. Used AWS CLI : To Upload Data in new created S3 Bucket (de-on-youtube-raw-useast1-dev14) Both JSON and CSV

```
upload: .\RU_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/RU_category_
id.json
PS E:\youtube> aws s3 cp . s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/ --recursive --exc
lude "*" --include "*.json"
>>
upload: .\RU_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/RU_category_
id.json
upload: .\CA_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/CA_category_
id.json
upload: .\MX_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/MX_category_
id.json
upload: .\US_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/US_category_
id.json
upload: .\IN_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/IN_category_
id.json
upload: .\GB_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/GB_category_
id.json
upload: .\DE_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/DE_category_
id.json
upload: .\JP_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/JP_category_
id.json
upload: .\FR_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/FR_category_
id.json
upload: .\KR_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/KR_category_
id.json
PS E:\youtube>
```

4. Created AWS Glue Catalog
    a. **What is AWS Catalog**



    b. **How AWS Glue Catalog Works**



c. **Created Role in IAM for Glue to access S3 in Crawler.**

Policy name
AmazonS3FullAccess
AWSGlueServiceRole
AWSGlueConsoleFullAccess

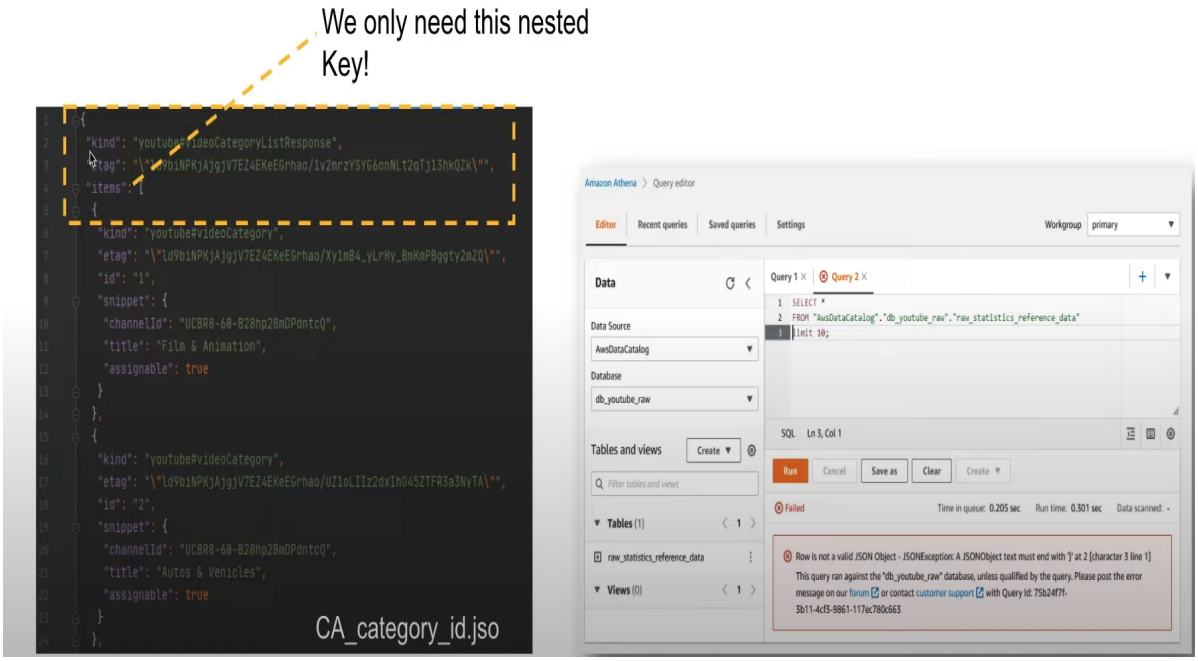
d. **Created a crawler to get metadata(Information about the table) in the raw dataset.**

Crawler properties

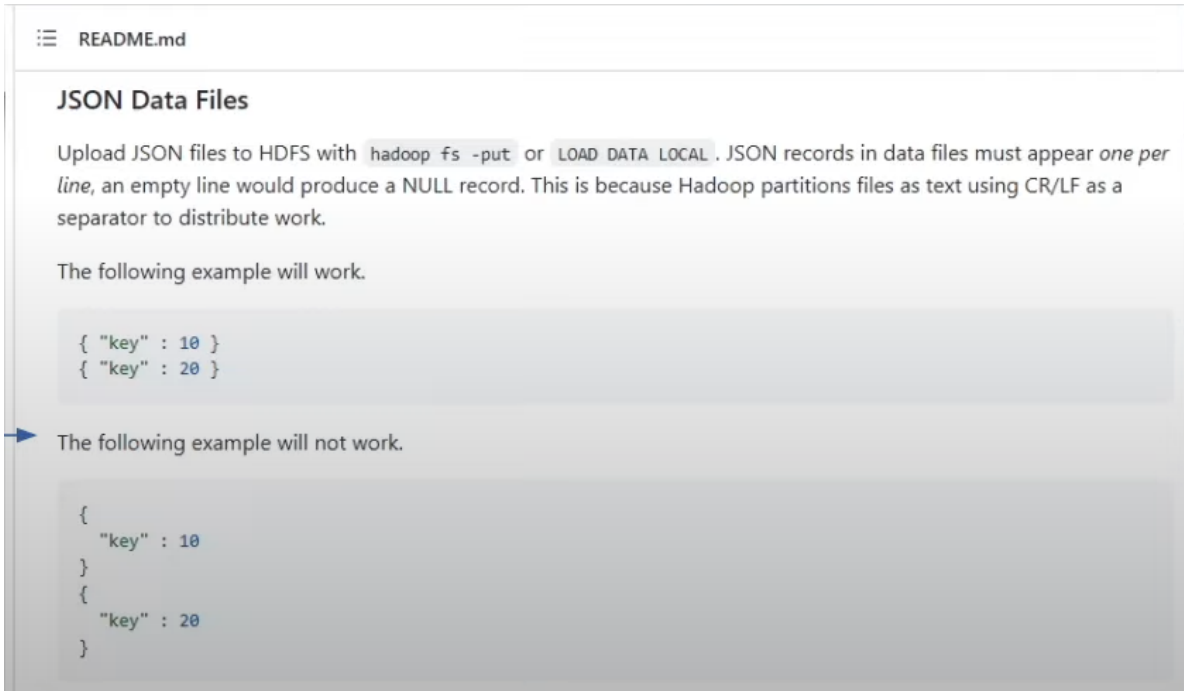
Name	IAM role
de-on-youtube-raw-glue-catalog-1	s3-glue-role
Description	**Security configuration**
-	-

e. **Tested Data in Athena**
   i. Created S3 Bucket to Store Athena Output.
   ii. Found Error in JSON Data Querying. (We only need Items from JSON).

iii.



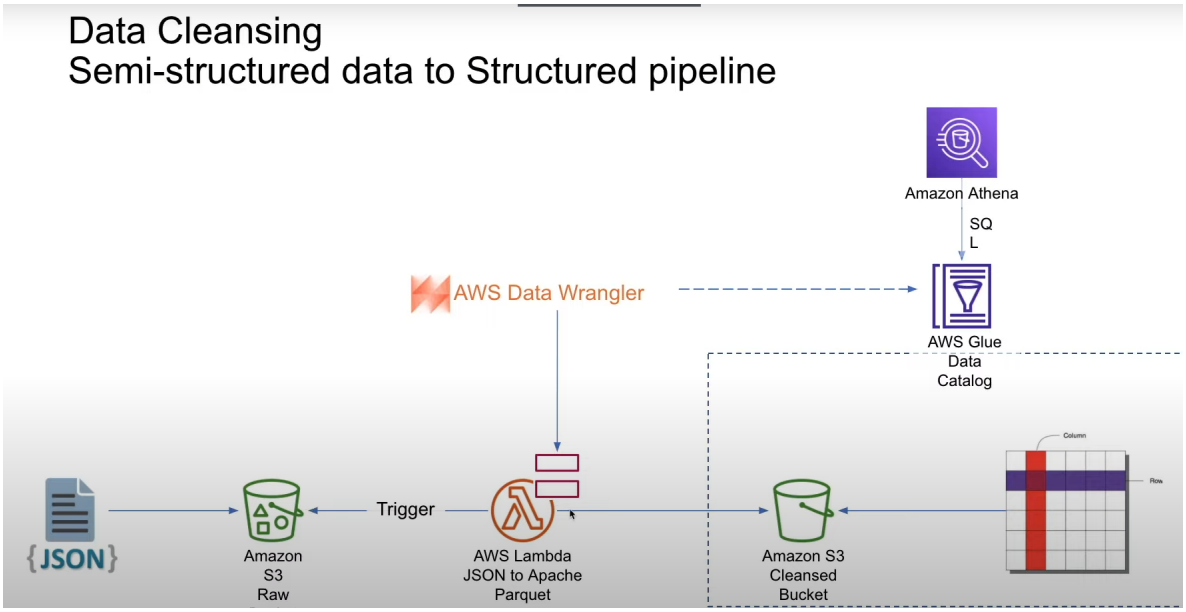
f. **Why?:**JSON records in data files must appear one per line, an empty file would produce a NULL record. This is because Hadoop partitions files as text using CR/LF as a separator to distribute work.



g. **Goal:** Data Cleansing

Create our light ETL: JSON to Apache Parquet.

- **Architecture for Data Cleaning semi structured to structured Pipeline using Lambda**



- **Created Spark code for Transformation in Lambda using AWS Wrangler.**

lambda_function

```
import awswrangler as wr
import pandas as pd
import urllib.parse
import os

# Temporary hard-coded AWS Settings; i.e. to be set as OS variable in Lambda
os_input_s3_cleansed_layer = os.environ['s3_cleansed_layer']
os_input_glue_catalog_db_name = os.environ['glue_catalog_db_name']
os_input_glue_catalog_table_name = os.environ['glue_catalog_table_name']
os_input_write_data_operation = os.environ['write_data_operation']


def lambda_handler(event, context):
    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'], encoding='utf-8')
    try:

        # Creating DF from content
        df_raw = wr.s3.read_json('s3://{}/{}'.format(bucket, key))

        # Extract required columns:
        df_step_1 = pd.json_normalize(df_raw['items'])

        # Write to S3
        wr_response = wr.s3.to_parquet(
            df=df_step_1,
            path=os_input_s3_cleansed_layer,
            dataset=True,
            database=os_input_glue_catalog_db_name,
            table=os_input_glue_catalog_table_name,
            mode=os_input_write_data_operation
```

5. Why we chose Lambda for Transformation?

- One of the AWS compute services
- Serverless
- High Available and scalable

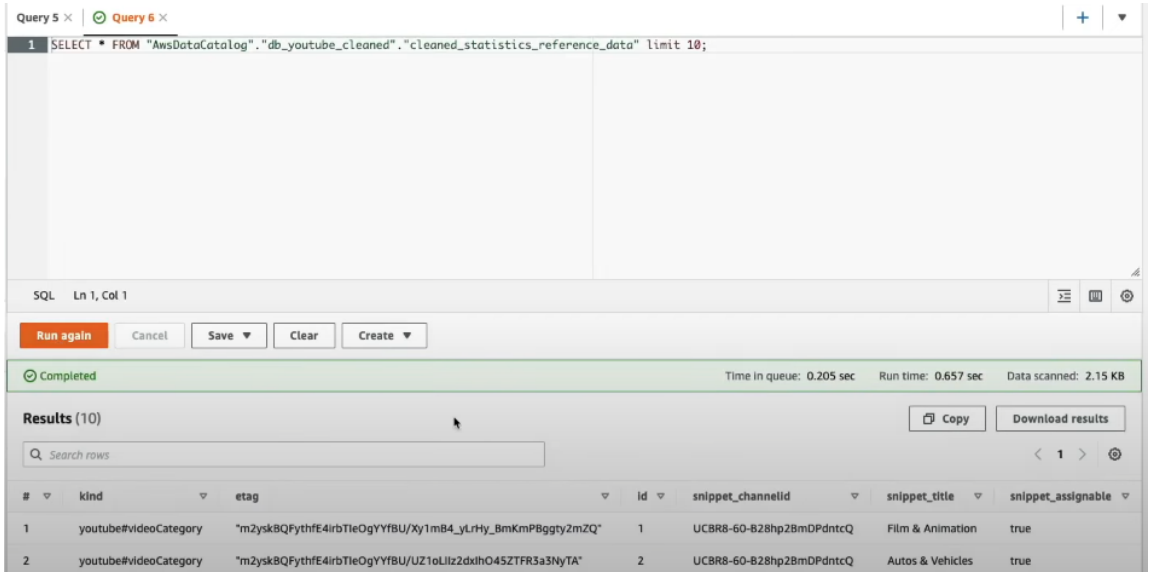
- Limits at this moment:
  - Deployment Package is 50 MB
  - 10 GB for Memory
  - 6 vCPUs
  - 15 min Timeout

Comparing the different data storage options

This table compares the characteristics of these four different data storage options for Lambda:

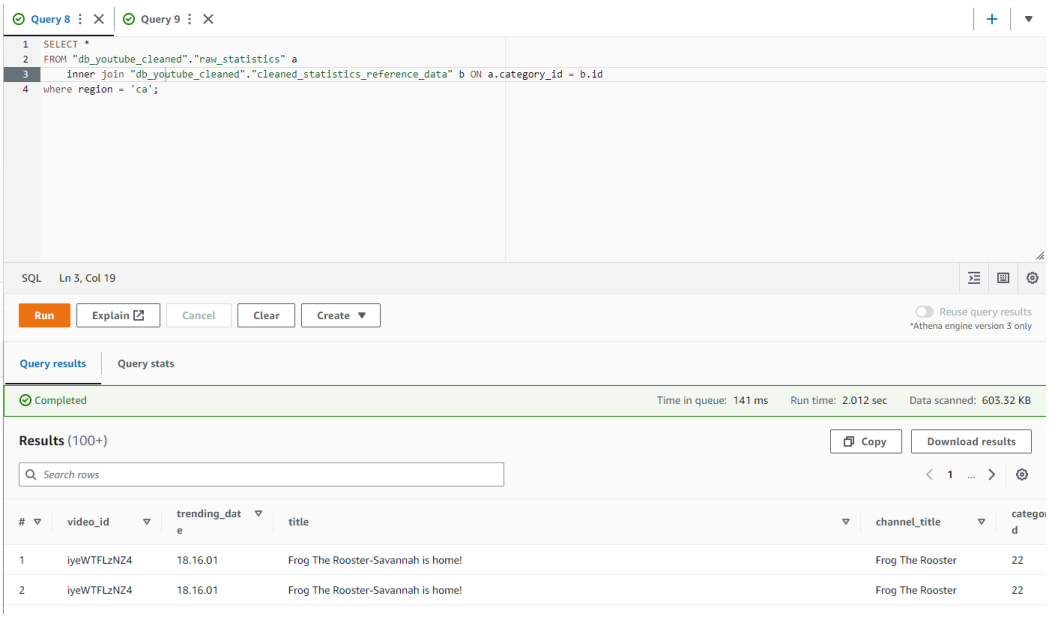
	Amazon S3	/tmp	Lambda Layers	Amazon EFS
Maximum size	Elastic	512 MB	50 MB (direct upload; larger if from S3).	Elastic
Persistence	Durable	Ephemeral	Durable	Durable
Content	Dynamic	Dynamic	Static	Dynamic
Storage type	Object	File system	Archive	File system
Lambda event source integration	Native	N/A	N/A	N/A
Operations supported	Atomic with versioning	Any file system operation	Immutable	Any file system operation
Object tagging	Y	N	N	N
Object metadata	Y	N	N	N
Pricing model	Storage + requests + data transfer	Included in Lambda	Included in Lambda	Storage + data transfer + throughput
Sharing/permissions model	IAM	Function-only	IAM	IAM + NFS
Source for AWS Glue	Y	N	N	N
Source for Amazon QuickSight	Y	N	N	N
Relative data access speed from Lambda	Fast	Fastest	Fastest	Very fast

- **Querying our cleaned Data in Athena .**



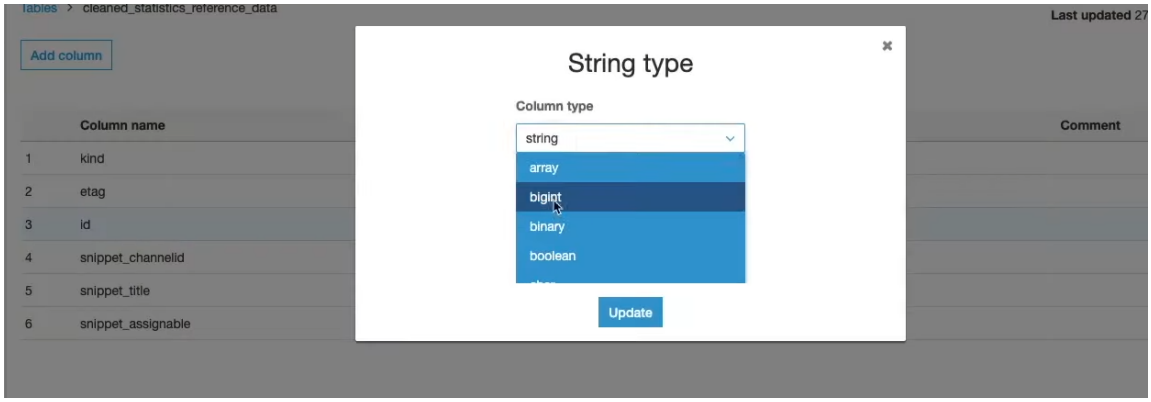
6. Joining JSON and CSV Data.

```
SELECT * FROM
 "db_youtube_cleaned"."raw_statistics" a
INNER JOIN
"db_youtube_cleaned"."cleaned_statistics_reference_data" b
ON a.category_id=b.id
;
```



7. Changed Data Type of Paraquet file output for efficient running of code.

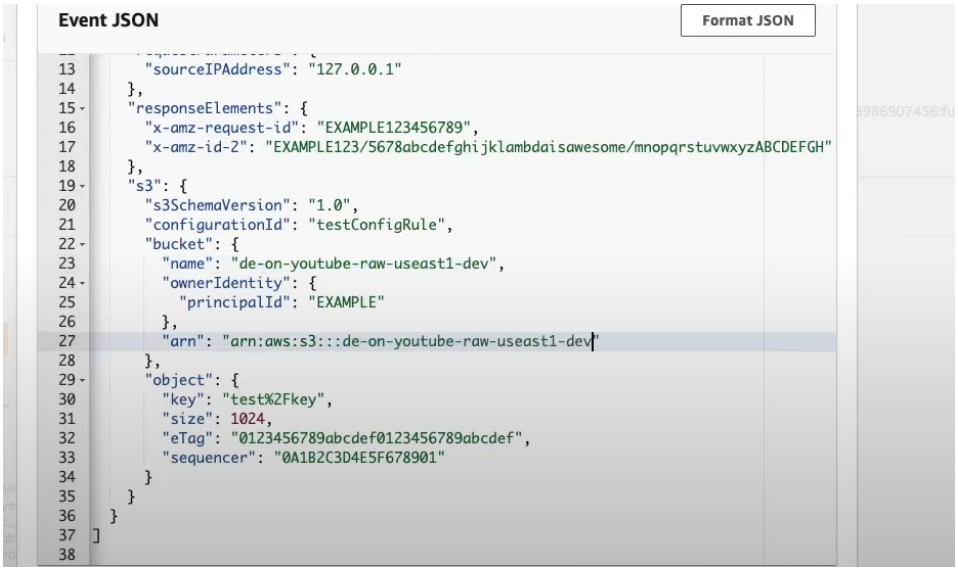
Steps: **1. Keep the data type change in the data catâlogue**



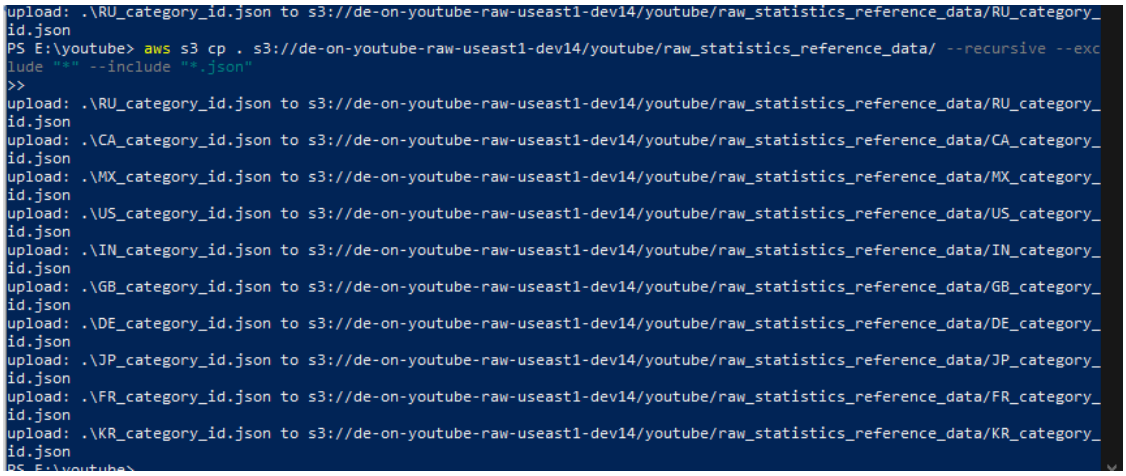
2. Delete our testing JSON file

3. Confirm APPEND in Lambda

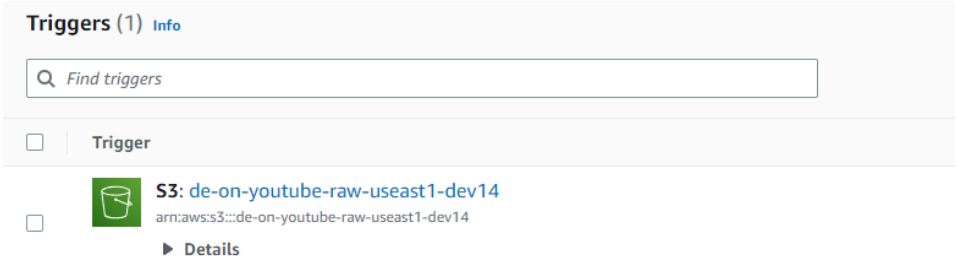
4. Run Test event in Lambda



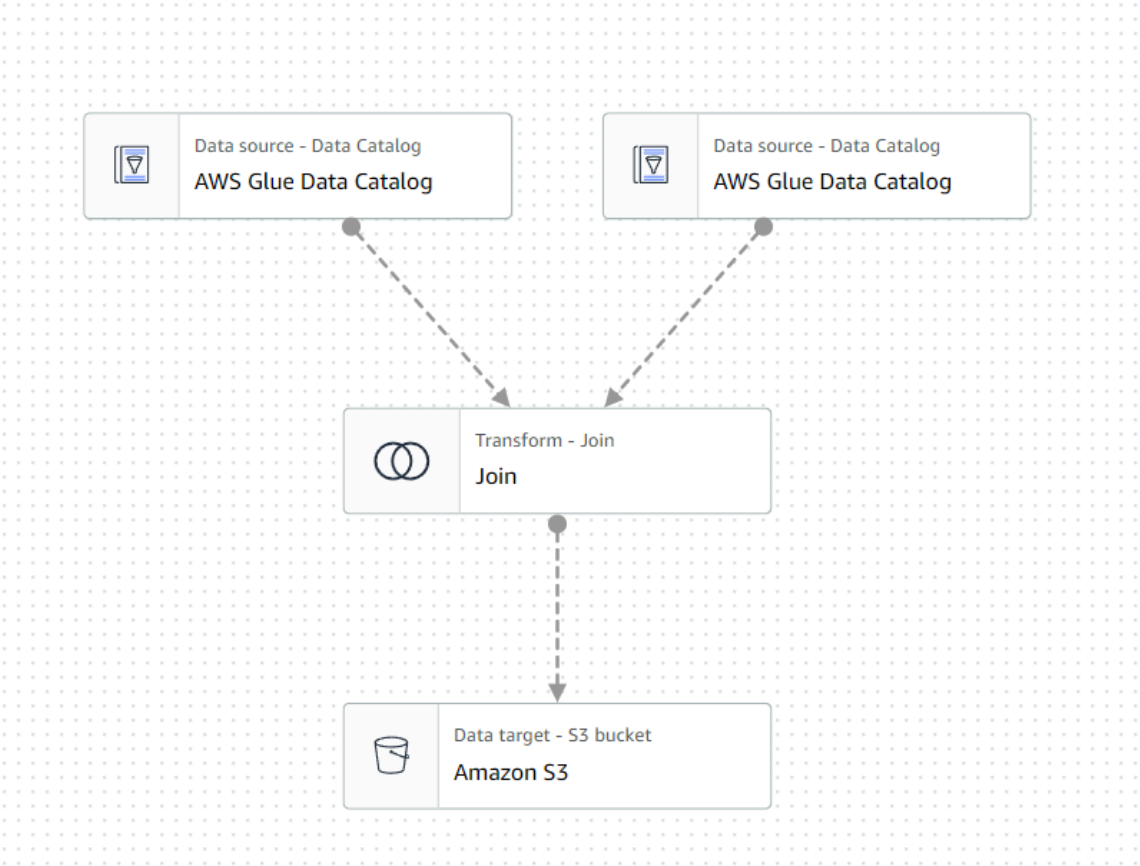
5. Copy again our data, from our laptops (AWS CLI)



6. Add the S3 Trigger to Lambda

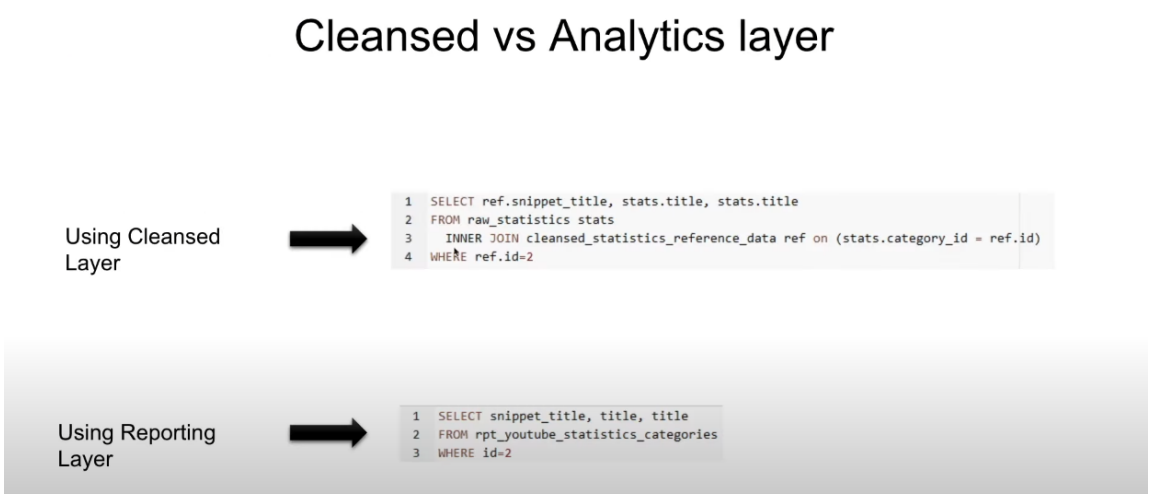


8. Created ETl Job for analytics



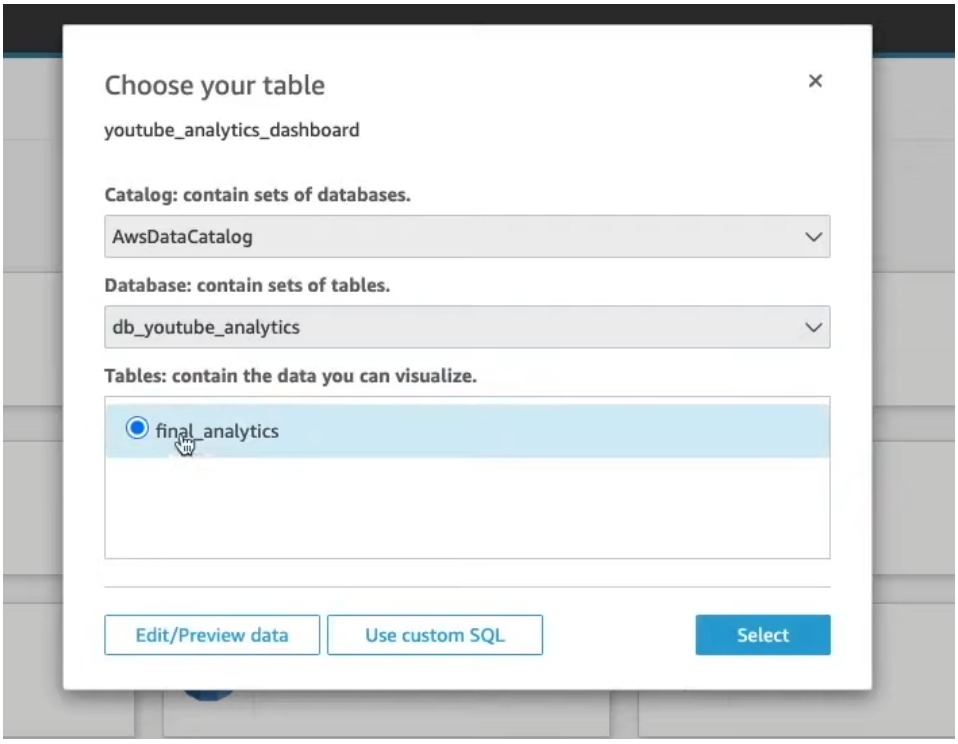
9. Build Reports based on Cleaned Data for Analysis

a. Cleansed vs Analytics Layer

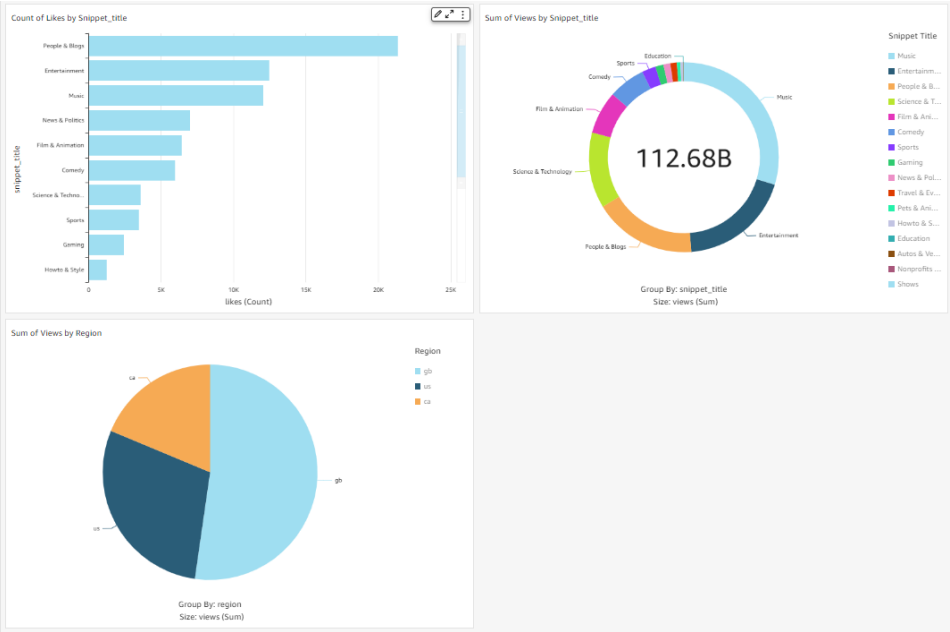


b. Dashboards

i. Connecting to Athena



ii. Building Reports



Learnings:

- The project involved extracting, cleaning, and transforming YouTube trending video data from around the world.
- Raw data was stored in an Amazon S3 bucket and cataloged using a Glue Crawler.
- Data cleaning and transformation was performed using Apache Spark.
- An ETL job was created using AWS Lambda to automate the data cleaning and transformation process.
- The cleaned data was queried using Amazon Athena and SQL to generate reports and insights.
- An ETL pipeline was created to copy the cleaned data to a separate S3 bucket for analysis.
- Amazon QuickSight was used to build various reports and visualizations based on the data.
- This project enabled the efficient and effective analysis of YouTube trending video data, resulting in the discovery of trends and patterns.