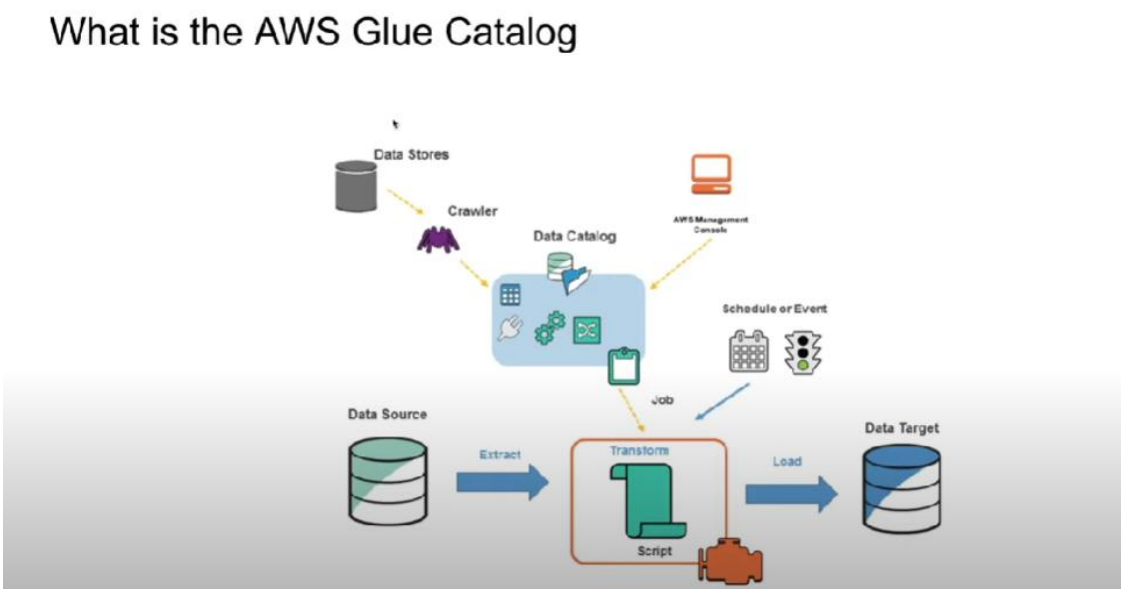
YOUTUBE DATA ANALYSIS

PROJECT OBJECTIVE

Get acquainted with the AWS tools to build out an End-to-End project using YouTube Data

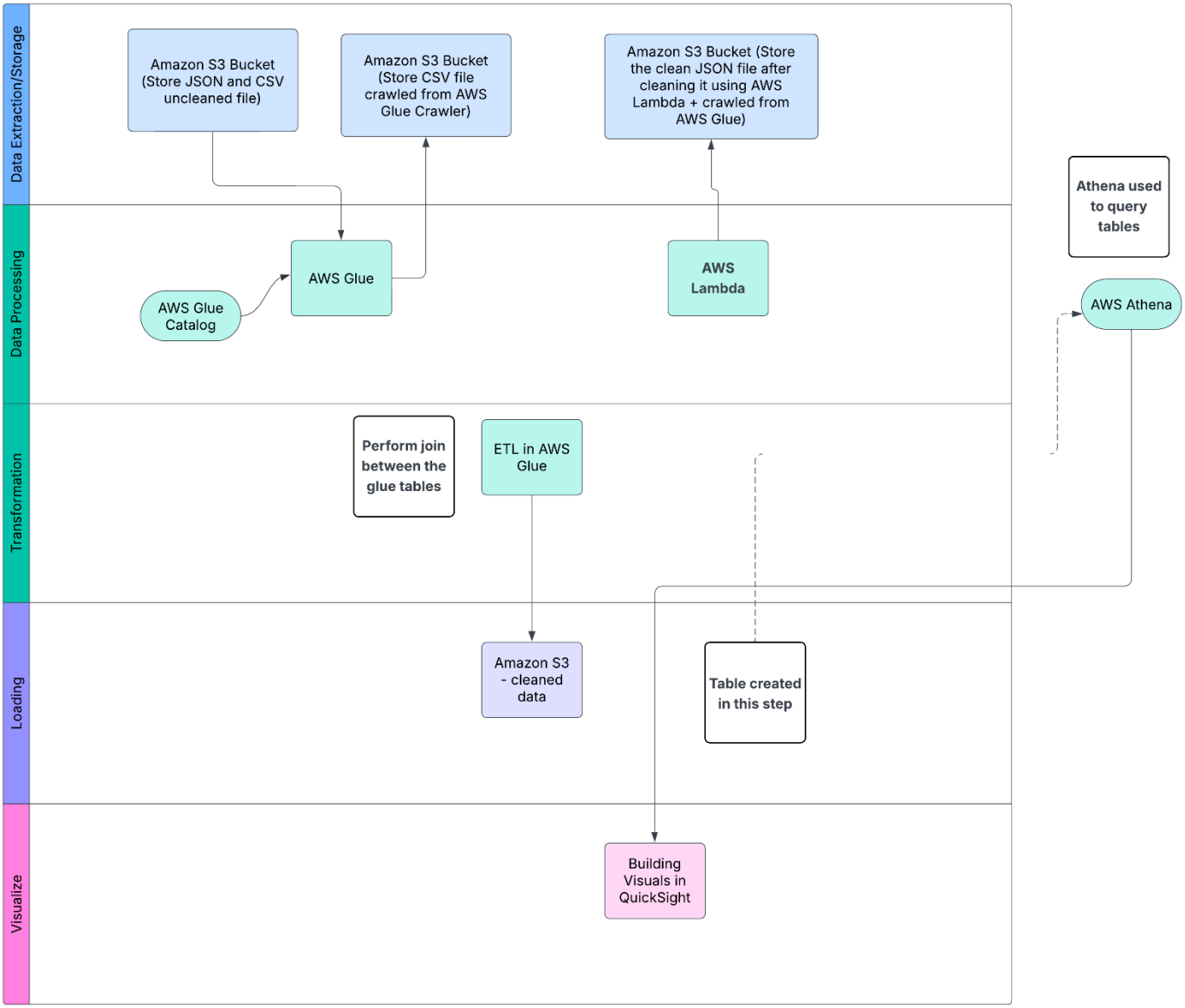
TECHNOLOGIES USED

1. **AWS CLI -** The AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.
2. **AWS S3 –** Created Buckets S3 stands for Simple Storage Service
3. **AWS Glue catalog –** It discovers the data from multiple sources, reads the metadata and the Glue Data Catalog serves as a central metadata repository. This data from AWS Glue Catalog can then be accessed for ETL and other analytics.



4. **AWS Athena – Used to Query the databased**
5. **AWS Lambda – Lambda function is triggered by an event. In this project, it is triggered by an S3 event**

PROJECT ARCHITECTURE



DATASET USED

Source - https://www.kaggle.com/datasets/datasnaek/youtube-new

This data includes daily trending YouTube videos.

This dataset includes several months (and counting) of data on daily trending YouTube videos. Data is included for the US, GB, DE, CA, and FR regions (USA, Great Britain, Germany, Canada, and France, respectively), with up to 200 listed trending videos per day.

EDIT: Now includes data from RU, MX, KR, JP and IN regions (Russia, Mexico, South Korea, Japan and India respectively) over the same time period.

Each region's data is in a separate file. Data includes the video title, channel title, publish time, tags, views, likes and dislikes, description, and comment count.

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. One such file is included for each of the five regions in the dataset.

STEPS INVOLVED

1. AWS account creation
    a. Created an AWS account with my email ID deshmukharpita728@gmail.com
2. Create Identity Access Management (IAM) user and IAM group
    a. Created a IAM User Arpita_Admin_User and assigned "AdministratorUser" policy to it. This user will have its own username (Arpita_Admin_User), password and a console login URL →https://339713111063.signin.aws.amazon.com/console
    b. Download AWS CLI

Using AWS CLI

C:\Windows\System32>aws

usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]

To see help text, you can run:

aws help

aws <command> help

aws <command> <subcommand> help

aws: error: the following arguments are required: command

---aws CLI has been installed

C:\Windows\System32>aws configure

AWS Access Key ID [********None]: AKIAU6GD2WQLXNTKUVX5**

AWS Secret Access Key [********5K5}]: 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**

Default region name [us-east-1]: us-east-1

Default output format [None]:

Command for viewing S3 bucket

aws s3 ls

3. Capture data from Kaggle
4. Create S3 bucket as the landing bucket – to store the data from Kaggle
    a. Follow a Naming Convention

```
s3://company-raw-awsregion-awsaccountID-
env/source/source_region/tablename/year=yyyy/
month=mm/day=dd/table_<yearmonthday>.<file_format>

env = dev, test, prod
source = name or indicator of source
source_region = region of data source
```

    b. Created a bucket with server encryption. S3 Bucket name **de-on-youtube-landing-useast1-dev**

5. Download the data from Kaggle website
    a. Go to the location of where the data is stored and enter *dir* command to view the files

```
:\Users\deshm\OneDrive\Personal Projects\Youtube Data Analysis\Data Source>dir
Volume in drive C is Windows-SSD
Volume Serial Number is 7A93-5814

Directory of C:\Users\deshm\OneDrive\Personal Projects\Youtube Data Analysis\Data Source

9/28/2024  09:36 PM    <DIR>          .
9/28/2024  09:38 PM    <DIR>          ..
9/28/2024  09:36 PM        64,067,991 CAvideos.csv
9/28/2024  09:36 PM             7,911 CA_category_id.json
9/28/2024  09:36 PM        63,040,138 DEvideos.csv
9/28/2024  09:36 PM             7,911 DE_category_id.json
9/28/2024  09:36 PM        51,424,708 FRvideos.csv
9/28/2024  09:36 PM             7,911 FR_category_id.json
9/28/2024  09:36 PM        53,213,441 GBvideos.csv
9/28/2024  09:36 PM             8,225 GB_category_id.json
9/28/2024  09:36 PM        59,600,439 INvideos.csv
```

6. Copy the data to S3 using AWS CLI – there are 2 ways to upload files from the local system to S3 – using the upload button on the S3 bucket or by using CLI
    a. Execute these commands in CLI to load the source data to S3 bucket

To copy all JSON Reference data to same location:

aws s3 cp . s3://de-on-youtube-landing-useast1-dev/youtube/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-landing-useast1-dev/youtube/raw_statistics/region=ca/

aws s3 cp DEvideos.csv s3://de-on-youtube-landing-useast1-dev/youtube/raw_statistics/region=de/

aws s3 cp FRvideos.csv s3://de-on-youtube-landing-useast1-dev/youtube/raw_statistics/region=fr/

aws s3 cp GBvideos.csv s3://de-on-youtube-landing-useast1-dev/youtube/raw_statistics/region=gb/

aws s3 cp INvideos.csv s3://de-on-youtube-landing-useast1-dev/youtube/raw_statistics/region=in/

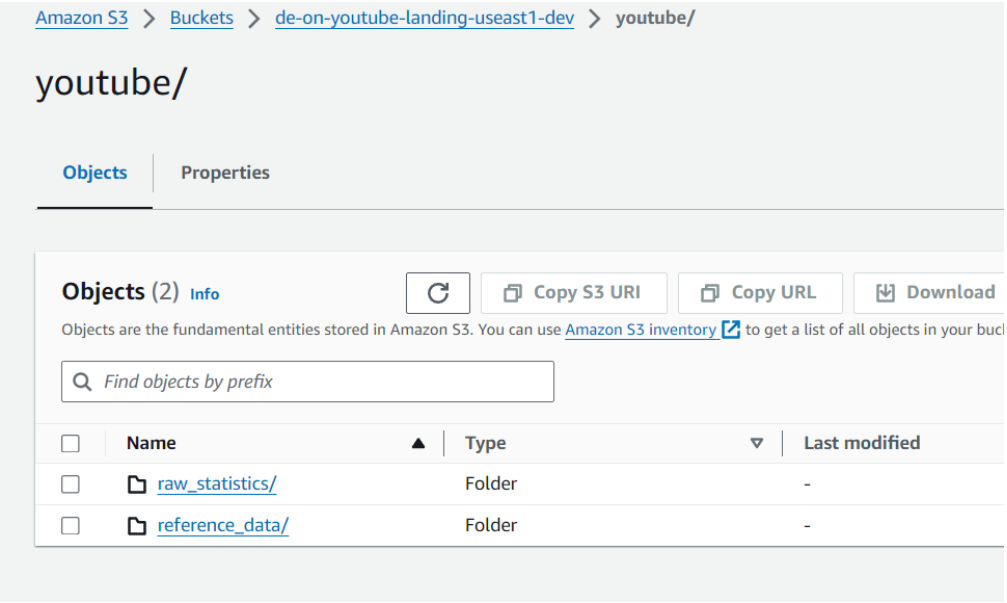
aws s3 cp JPvideos.csv s3://de-on-youtube-landing-useast1-dev/youtube/raw_statistics/region=jp/

aws s3 cp KRvideos.csv s3://de-on-youtube-landing-useast1-dev/youtube/raw_statistics/region=kr/

aws s3 cp MXvideos.csv s3://de-on-youtube-landing-useast1-dev/youtube/raw_statistics/region=mx/

aws s3 cp RUvideos.csv s3://de-on-youtube-landing-useast1-dev/youtube/raw_statistics/region=ru/

aws s3 cp USvideos.csv s3://de-on-youtube-landing-useast1-dev/youtube/raw_statistics/region=us/

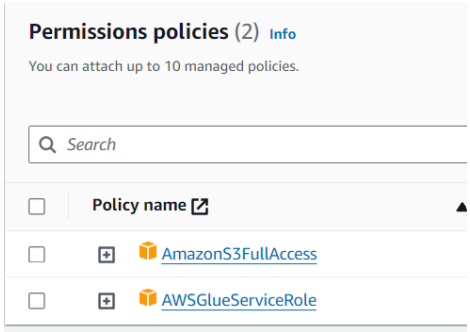


7. Go to AWS Glue > Data Catalog > Crawlers
    a. Create a Crawler – this will include assigning a crawler name, assigning the S3 bucket reference_data path as the data source for the crawler, create a IAM role so that AWS Glue can access the S3.

        Crawler Name → de-on-youtube-landing-glue-catalog-1

    b. Create IAM role and assign permissions to that IAM glue role to access S3.

        Role name glue-accessS3



    c. While creating a crawler you will have to add a database where you want the crawler to crawl the source data i.e., S3 bucket and output the data in the database

        Database name → de-youtube-database-raw

Name	Description	Tags
de-on-youtube-landing-glue-catalog-1	glue crawler for youtube JSON files	-

Step 2: Choose data sources and classifiers Edit

Data sources (1) Info

The list of data sources to be scanned by the crawler.

Type	Data source	Parameters
S3	s3://de-on-youtube-landing-useast1-...	Recrawl new only

Step 3: Configure security settings Edit

Configure security settings

IAM role	Security configuration	Lake Formation configuration
glue-accessS3	-	-

Step 4: Set output and scheduling Edit

Set output and scheduling

Database	Table prefix - *optional*	Maximum table threshold - *optional*	Schedule
de-youtube-database-raw	-	-	On demand

d. After creating the crawler, select the crawler and hit “RUN CRAWLER”

AWS Glue > Crawlers > de-on-youtube-landing-glue-catalog-1

de-on-youtube-landing-glue-catalog-1

Last updated (UTC) September 29, 2024 at 04:31:14 | Run crawler | Edit | Delete

Crawler properties

Name	IAM role	Database	State
de-on-youtube-landing-glue-catalog-1	glue-accessS3	de-youtube-database-raw	READY
Description	**Security configuration**	**Lake Formation configuration**	**Table prefix**
glue crawler for youtube JSON files	-	-	-
Maximum table threshold			
-			

▶ Advanced settings

Crawler runs | Schedule | Data sources | Classifiers | Tags

Crawler runs (1)

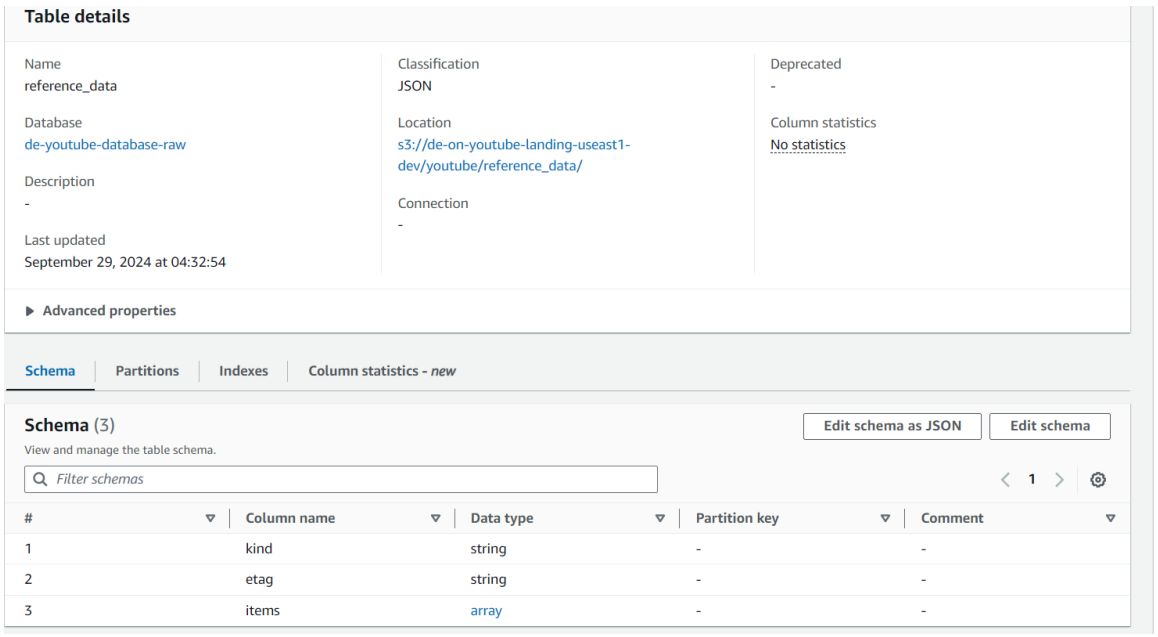
The list of crawler runs for this crawler.

Stop run | View CloudWatch logs | View run details

Filter data | Filter by a date and time range | 1

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
September 29, 2024 at 04:31	-	47 s	Running	-	-

e. The crawler created this table which basically includes the schema of the data crawled from S3



8. Now to view the data click on "Table data", the system will redirect to AWS Athena – created a S3 bucket to store the query results

S3 bucket name - de-on-youtube-landing-useast1-athena

NOTE: To query in AWS Athena you need to provide an output location i.e. S3 bucket

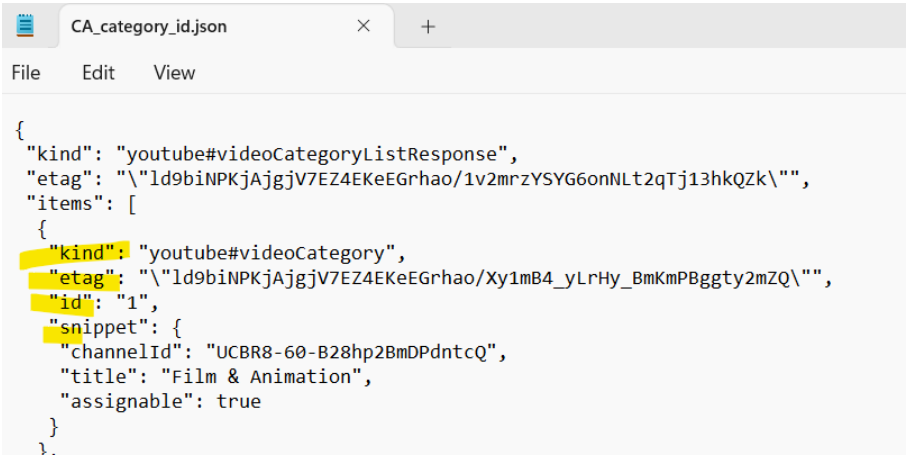
The query gave an error

SELECT * FROM "AwsDataCatalog"."de-youtube-database-raw"."reference_data" limit 10;

AWS Glue wants the key value pair in certain format

"AWS Athena does not support multi-line JSON.
Athena knowledge center
Make sure your JSON record is on a single line
Athena doesn't currently support multi-line JSON records."

The current JSON file is structures like this: multi-line JSON




```
{
 "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
   }
  },
```

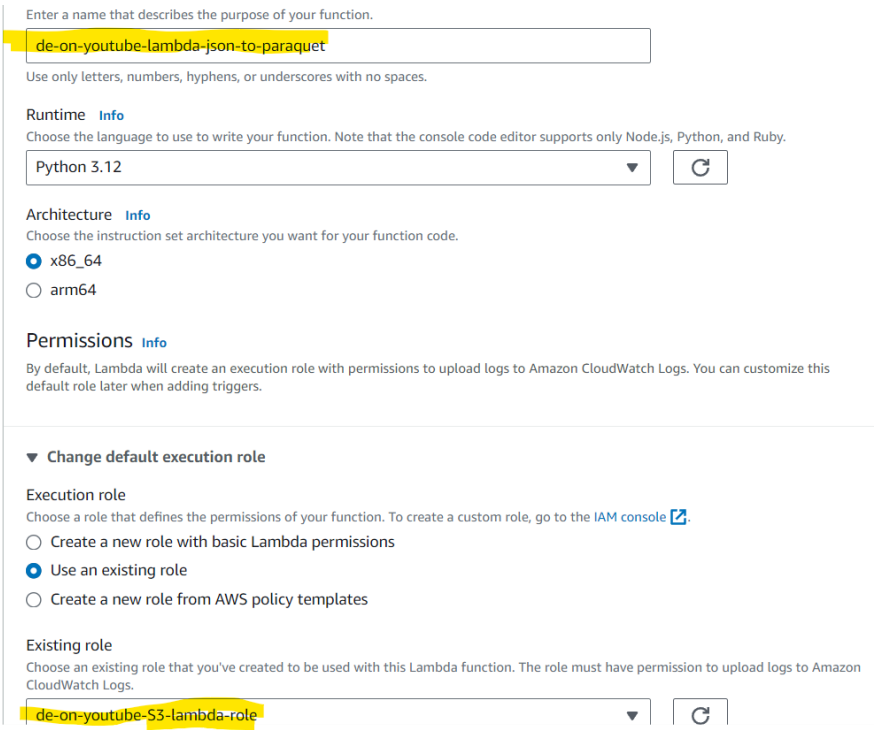
Solution: Need to convert the JSON file to a structured format i.e. to Apache Parquet format but only need to extract inside items

***** what is Apache paraquet?**

9. Using AWS Lambda to convert JSON to paraquet – AWS Lambda supports Python. It can be used to clean the JSON file and convert to Paraquet file into a new S3 bucket which includes the cleansed data and re-use the AWS Glue Crawler to use the cleansed data.
    a. Create a Lambda function – Also need to create a role for lambda function to access the AWS services like S3

    Lambda function name – de-on-youtube-lambda-json-to-paraquet

    Role created for lambda to access S3 - de-on-youtube-S3-lambda-role



    b. Python Code written in the lambda function just created – this function gets triggered by the S3 event.

    This S3 event, pulls from the main S3 landing bucket and only pulls the CA JSON. So, the cleansed Glue data table will only have CA region Paraquet file but all the regions CSV files. *(All JSON files are same from the data source)*

Event name

Test_s3_put

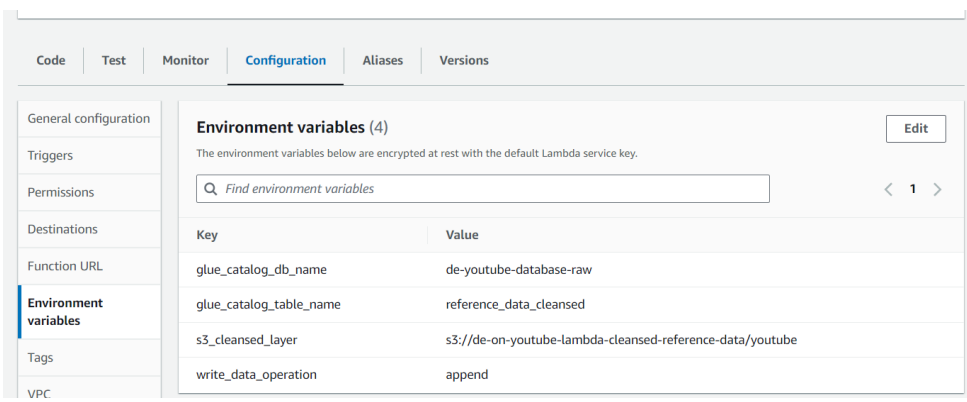
Event JSON


```
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
        "x-amz-request-id": "EXAMPLE123456789",
        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",
        "bucket": {
          "name": "de-on-youtube-landing-useast1-dev",
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::de-on-youtube-landing-useast1-dev"
        },
        "object": {
          "key": "youtube/reference_data/CA_category_id.json",
          "size": 1024,
          "eTag": "0123456789abcdef0123456789abcdef",
          "sequencer": "0A1B2C3D4E5F678901"
        }
```


```
import awswrangler as wr

import pandas as pd

import urllib.parse

import os


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'])
```

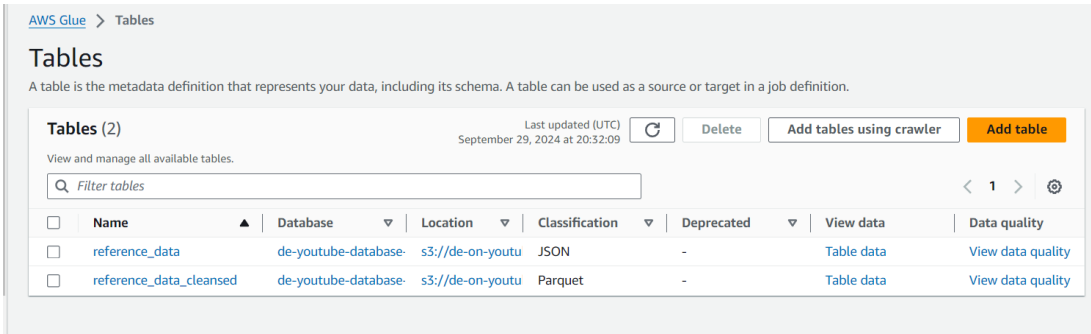
```
    # Write to S3
    wr_response = wr.s3.to_parquet(
        df=df_step_1,
        path=os_input_s3_cleansed_layer,
        dataset=True,
        database=os_input_glue_catalog_db_name,
        table=os_input_glue_catalog_table_name,
        mode=os_input_write_data_operation
    )

    return wr_response
  except Exception as e:
    print(e)
    print('Error getting object {} from bucket {}. Make sure they exist and your bucket is in the same region as this function.'.format(key, bucket))
    raise e
```

c. Create the environment variables. Go to Configuration>Environment variables and hit Add → add the database name and data table name which was generated from the AWS Glue (Here I used the same database name but used a different table name to view the difference in data)



d. The cleansed data will go to this S3 bucket → de-on-youtube-lambda-cleansed-reference-data Create this S3 bucket

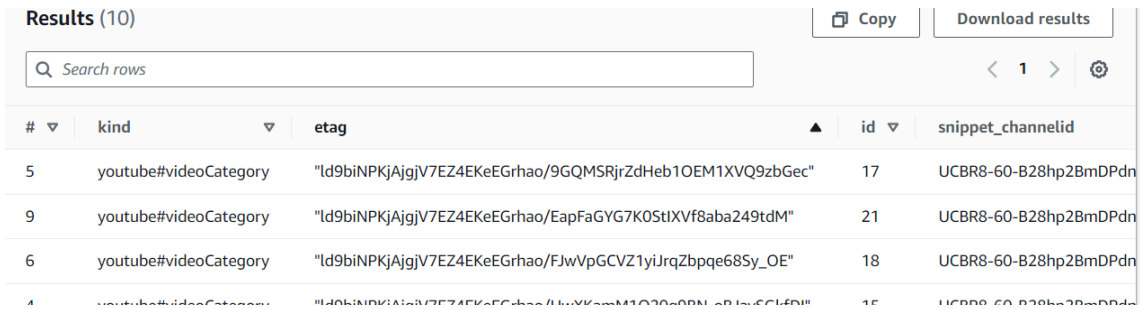
e. Deploy and test the Python code. Refresh the Glue tables



f. Add a layer if there is a package error

g. If you run this query in AWS Athena – it should give the results

SELECT * FROM "AwsDataCatalog"."de-youtube-database-raw"."reference_data_cleansed" limit 10;



STEPS RECAP

i. Downloaded data from Kaggle (includes reference data in JSON and region wise data in CSV)
ii. Built a S3 "landing" bucket to collect the data
iii. Created AWS Glue crawler to capture the JSON data from the above created S3 bucket and create catalogue out of it. This crawler generates a database name and a table with the data *(Created an AWS Glue Role to access the S3 bucket)*
iv. Click on the data table to view the query the data in AWS Athena. If this step gives an error, utilize AWS Lambda to clean it
v. Created AWS lambda function to cleanse the JSON format data and converted it to Paraquet. Wrote a Python script to perform the cleaning. *(Created a lambda role to access S3 and Glue services)*
vi. Deployed and tested the function to generate a paraquet file in the AWS Glue database but a different table. This data is also stored in a "Cleansed" S3 bucket
vii. The cleansed data can be viewed when queried on the newly generated data table (in Glue) in AWS Athena.

10. Create another crawler to grab the CSV files into S3 bucket – new crawler, same role, same database

Name	Description	Tags
de-on-youtube-CSV-crawler	This crawler grabs the CSV data from the data source	-

Step 2: Choose data sources and classifiers Edit

Data sources (1) Info

The list of data sources to be scanned by the crawler.

Type	Data source	Parameters
S3	s3://de-on-youtube-landing-useast1-...	Recrawl new only

Step 3: Configure security settings Edit

Configure security settings

IAM role	Security configuration	Lake Formation configuration
glue-accessS3	-	-

Step 4: Set output and scheduling Edit

Set output and scheduling

Database	Table prefix - *optional*	Maximum table threshold - *optional*	Schedule
de-youtube-database-raw	-	-	On demand

All the CSV data is now stored in raw_statistics

Tables (3)

Last updated (UTC)
September 30, 2024 at 03:02:59

View and manage all available tables.

Filter tables

Name	Database	Location	Classification
raw_statistics	de-youtube-database-	s3://de-on-youtube-la	CSV
reference_data	de-youtube-database-	s3://de-on-youtube-la	JSON
reference_data_cleansed	de-youtube-database-	s3://de-on-youtube-la	Parquet

11. Performing a JOIN to join between the JSON cleansed data and CSV data.

    Faced an error where category_id from CSV is bigint but JSON id is string. So, I switched the datatype of JSON Id in the Glue table to int from string. Changing the data type in the data table in AWS glue does not change the type in Paraquet file in S3 bucket. Don't use cast operation in the query but update the schema so that data types match.

    How to resolve this error?

i. Delete the paraquet object created in the S3 bucket
ii. Rerun the lambda function

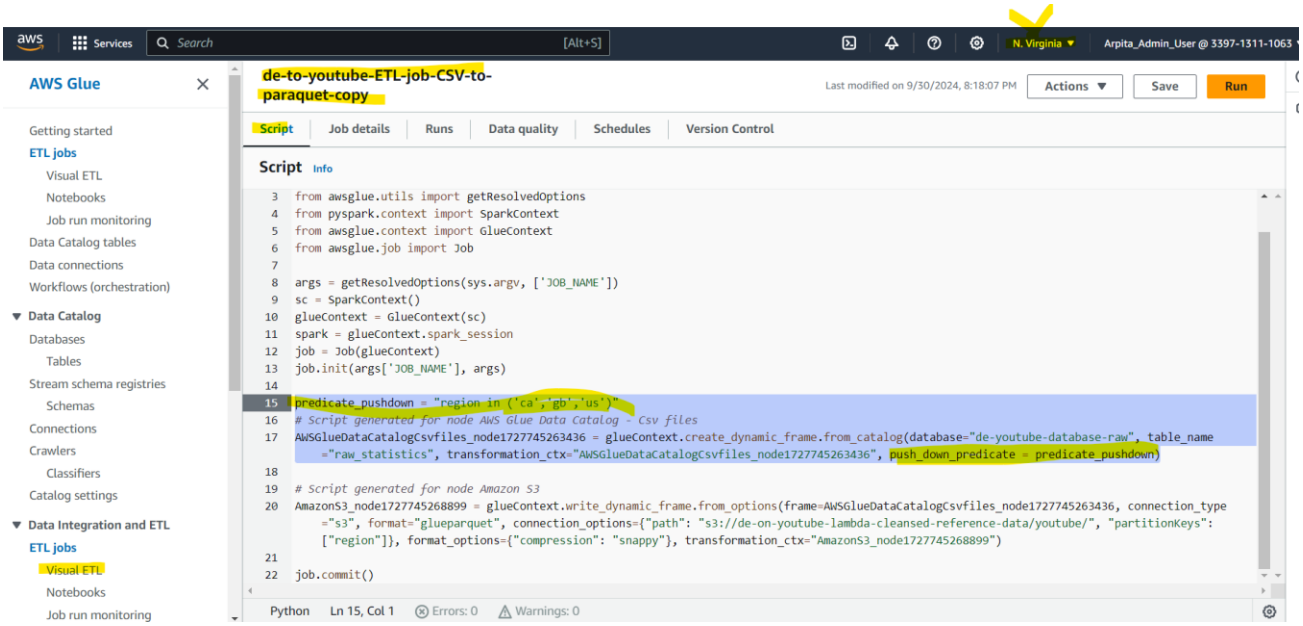
A new paraquet object file will appear in the S3 bucket with the Glue table schema changes because of the append operation. When the S3 object is deleted, the cleansed JSON data table in Glue is also empty. After running the Lambda function, the data in the cleansed table reappears

12. Create a Glue ETL Job to pull the CSV files from the Glue Data Catalog Raw statistics to a the Cleansed S3 bucket so that all the data source files are in one cleansed S3 bucket. The CSV files will be stored in paraquet format in the cleansed s3 bucket

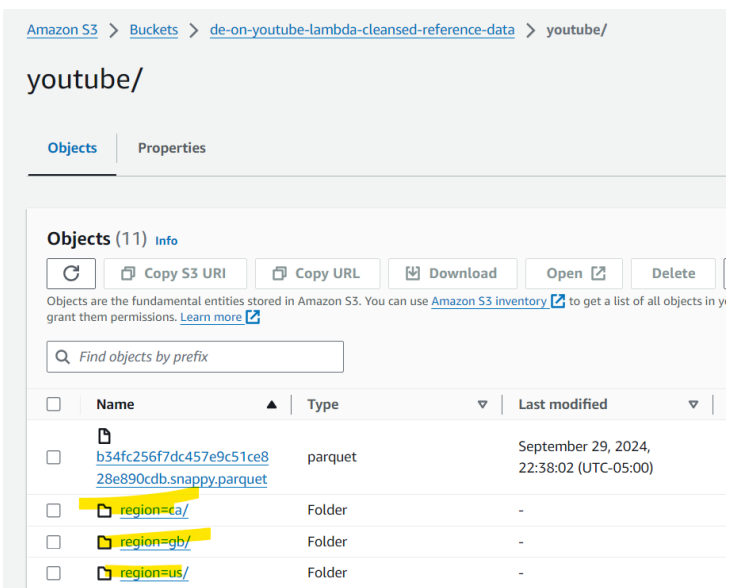
Faced an issue: I was trying to create the ETL job with source as the Glue Data Catalog but the visual ETL wouldn't show me the database and data table I created in the catalog. The region was switched from Virginia to Ohio and that is the reason it couldn't read the tables

The visual ETL can be created using the drag and drop feature in Glue ETL. If there is a need to edit the script, it is suggested that the ETL job has a copy created because the visual goes away.

--Added push_down_predicate to filter the Glue table

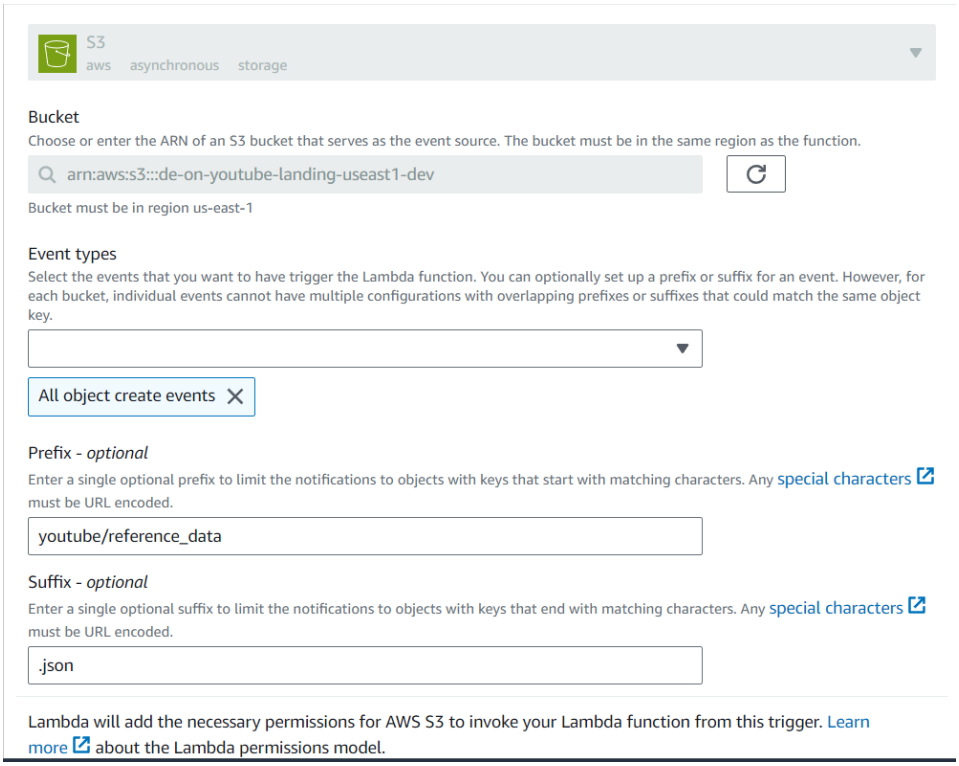


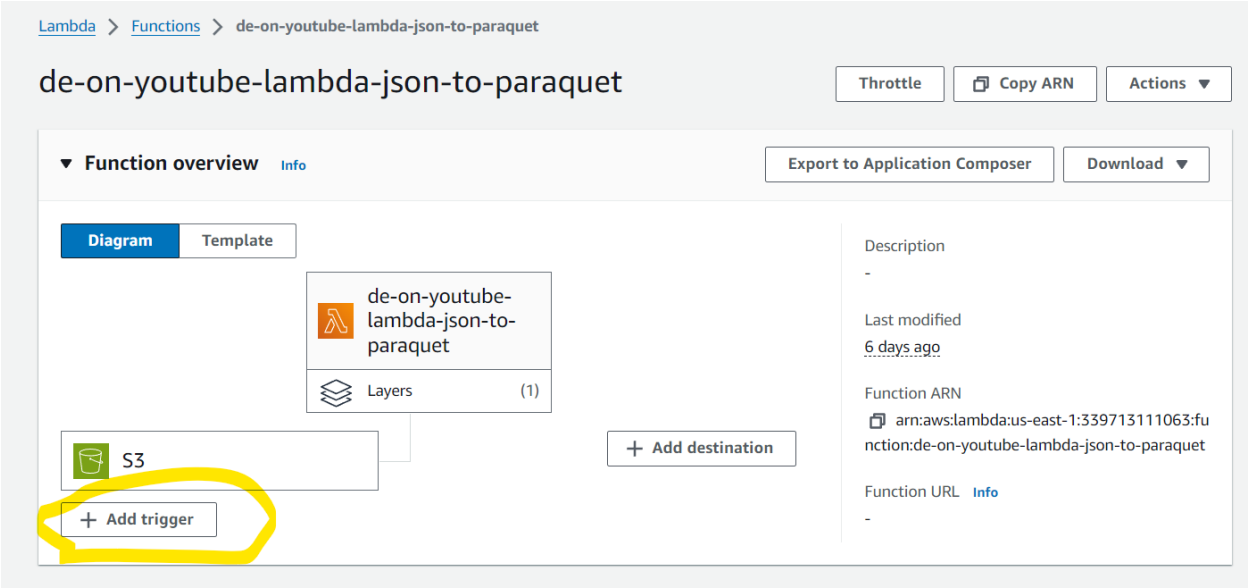
This created 3 folders with the 3 regions and inside this 3 regions is the paraquet files



13. Creating a Trigger to kick off Lambda function - Earlier only one JSON file was uploaded i.e. CA JSON file from the S3 test event. Uploading other JSON files by using a trigger. This trigger will kick off the lambda function if there is any new JSON file in the landing S3 bucket
    a. First delete the JSON files from the landing S3 bucket

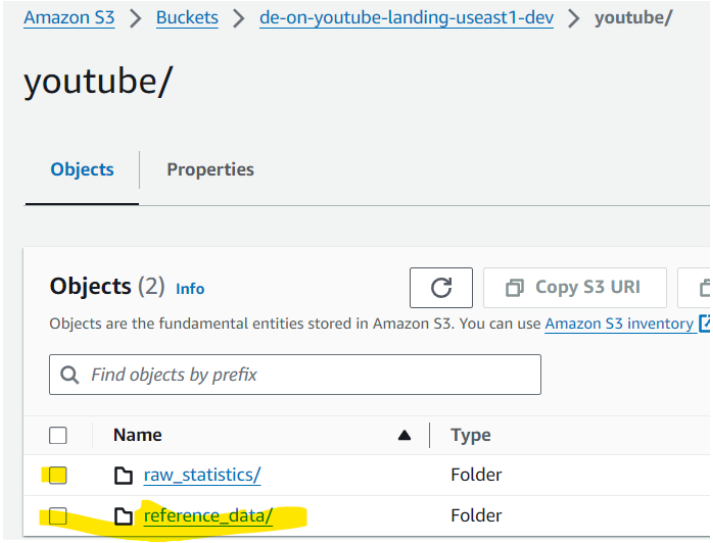
b. Create the below trigger





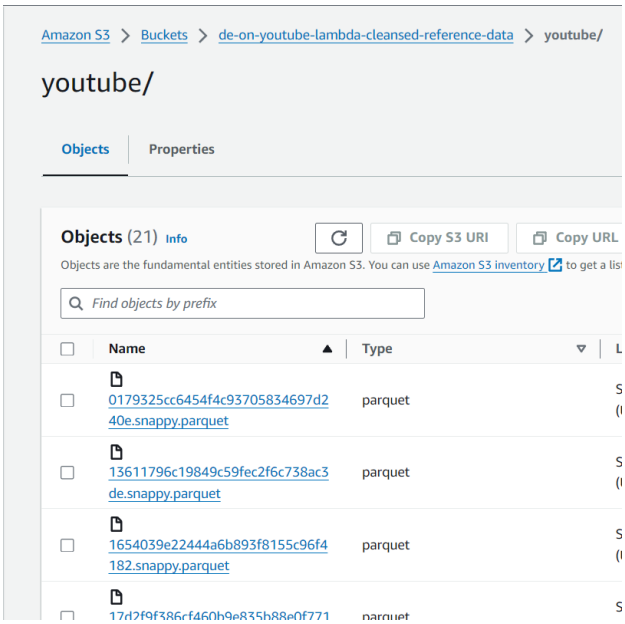
c. Rerun the commands using AWS CLI where all the JSON files gets uploaded to the S3 bucket

```
C:\Users\deshm\OneDrive\Personal Projects\Youtube Data Analysis\Data Source>aws s3 cp . s3://de-on-youtube-landing-useast1-dev/youtube/reference_data/ --recursive --exclude "*" --include "*.json"
```

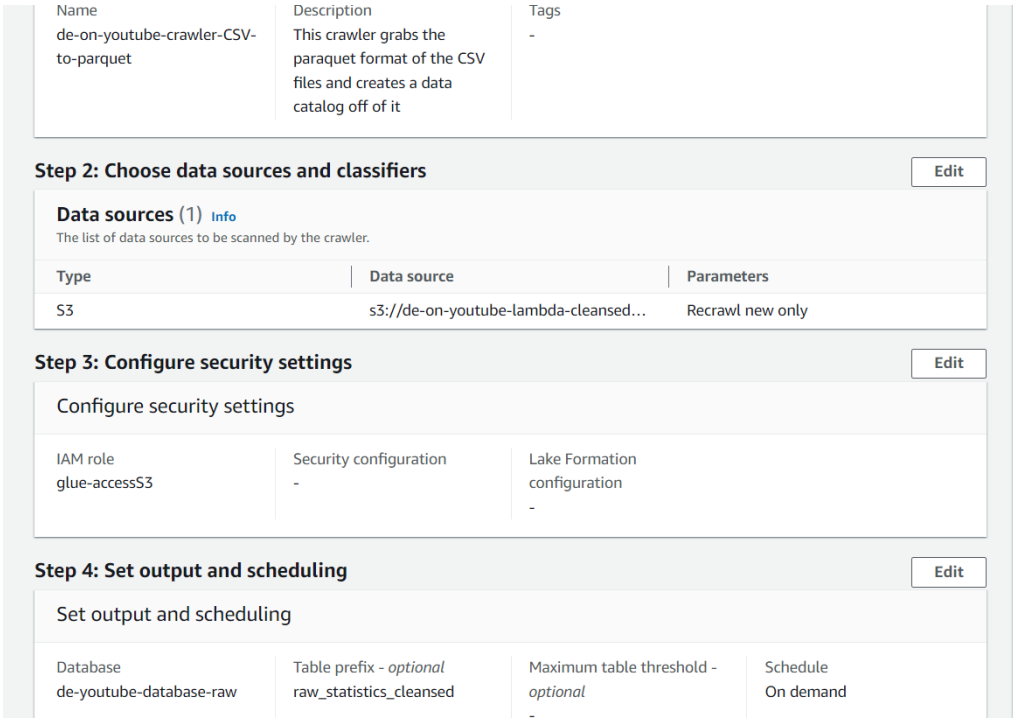


This JSON files added will automatically kick off the lambda function

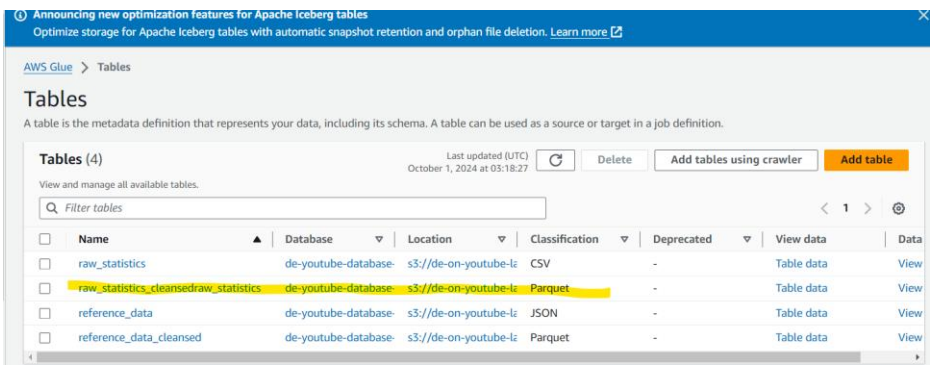
Below all 10 JSON files get added to the cleansed S3 bucket – one for each JSON



14. Create another crawler to create a data catalog of the CSV files converted to parquets using the ETL visual. This ETL visual tool, at output format was selected as paraquet.



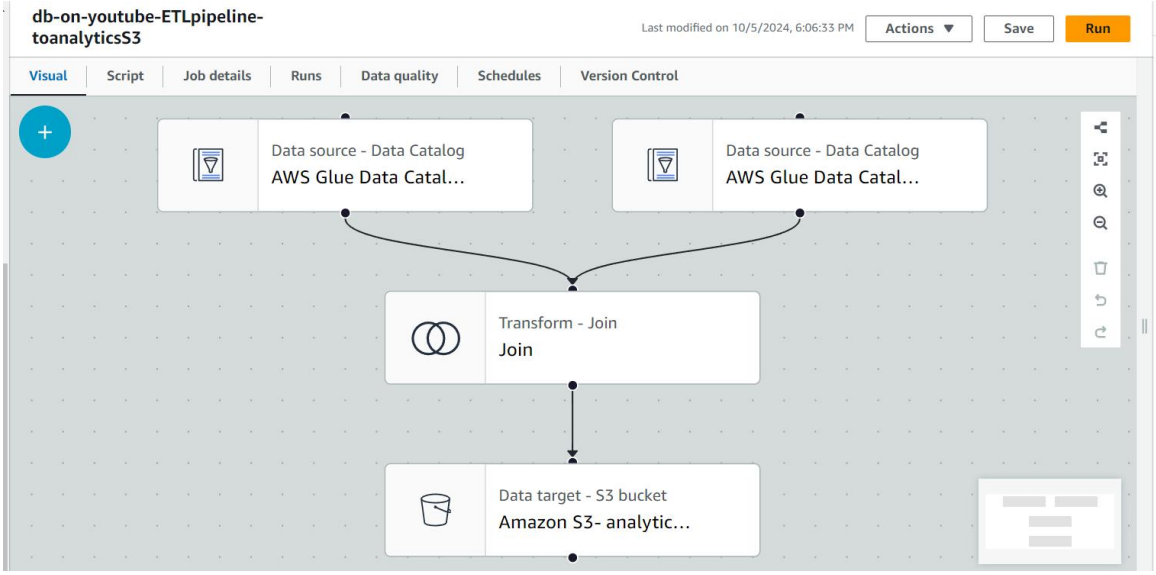
This crawler will create a Glue data table called raw_statistics_cleansed



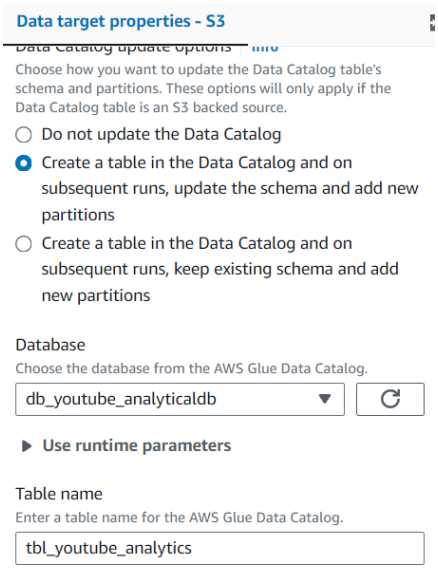
15. Next step is creating the S3 bucket which will include the join between both the cleaned data tables. The below query should be automated using a ETL pipeline in AWS Glue because the query shouldn't be written again and again so building an ETL pipeline from cleansed S3 bucket to another S3 bucket specifically for reporting

```
select *

from raw_statistics_cleansedraw_statistics csv

join reference_data_cleansed json on csv.category_id = json.id
```

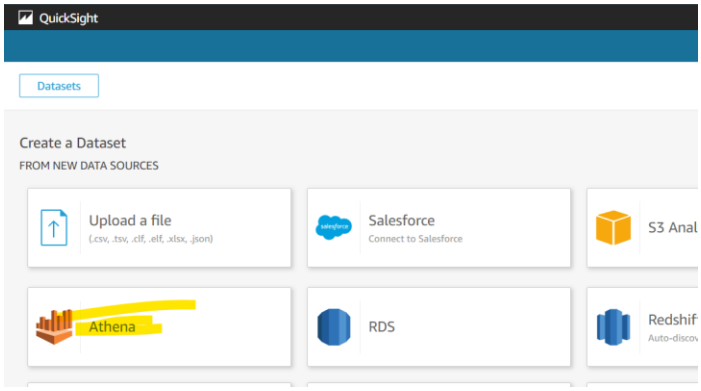
NOTE: The ETL pipeline can be built with target as a AWS table in already exisiting database as well but wanted to create a S3 bucket for the final data and create a different analytical database along with the table



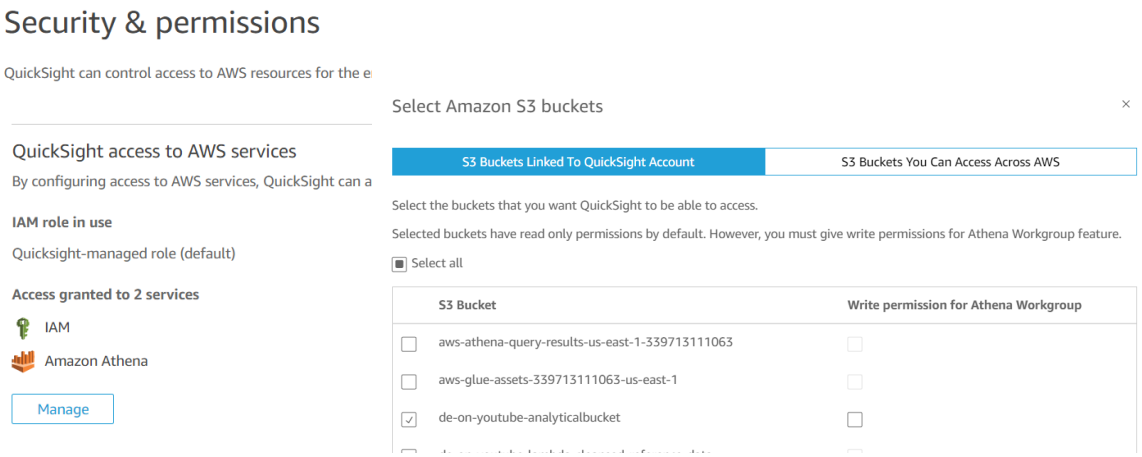
Created a partition based on region in the target S3 bucket. Automatically created the AWS Glue database and data table



16. Visualization in AWS Quicksight\
    a. Created a QuickSight account
    b. Create a dataset and connect it to Athena – db_youtube_analyticaldb



c. Click on your profile and go to manage quicksight and assign permission to S3 bucket de-on-youtube-analyticaldb under security and permissions



Resources used:

1. https://aws.amazon.com/cli/