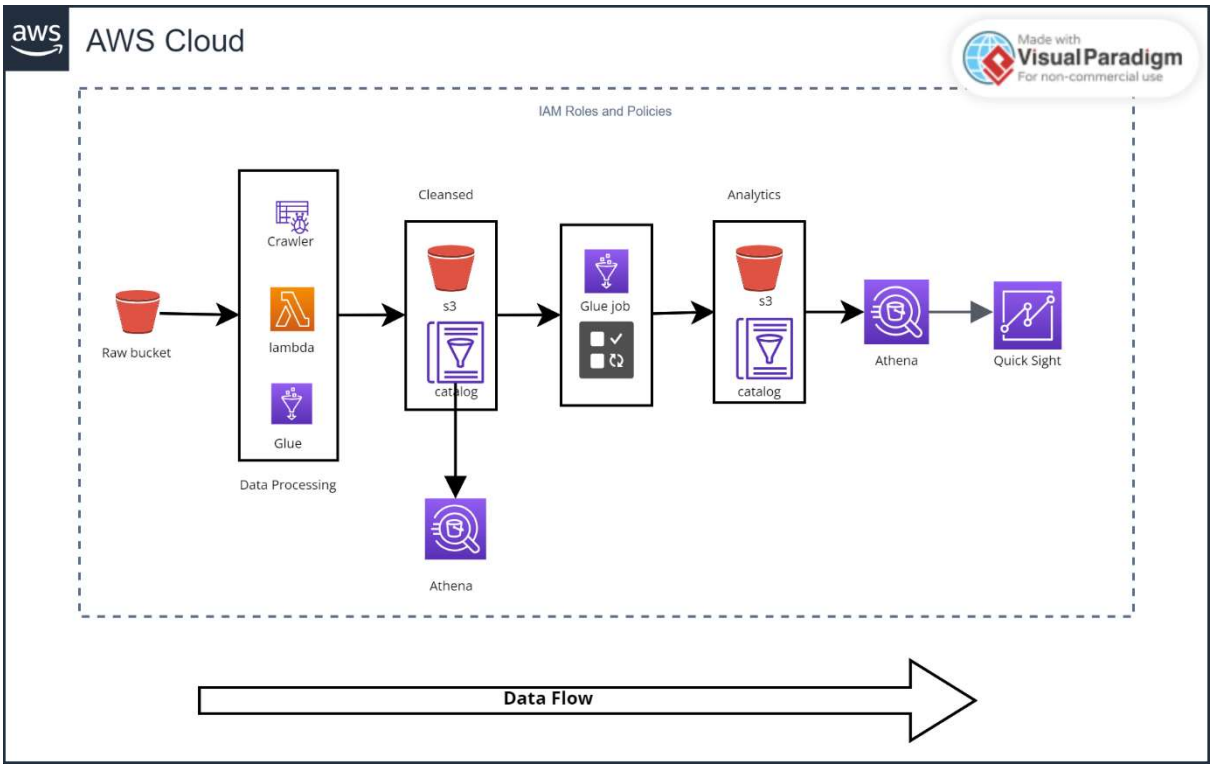
Data Analysis and Visualisation on Trending YouTube Data using AWS Glue and QuickSight

Objective : To perform Data Analysis and Visualisation on the trending youtube data using various AWS Services.

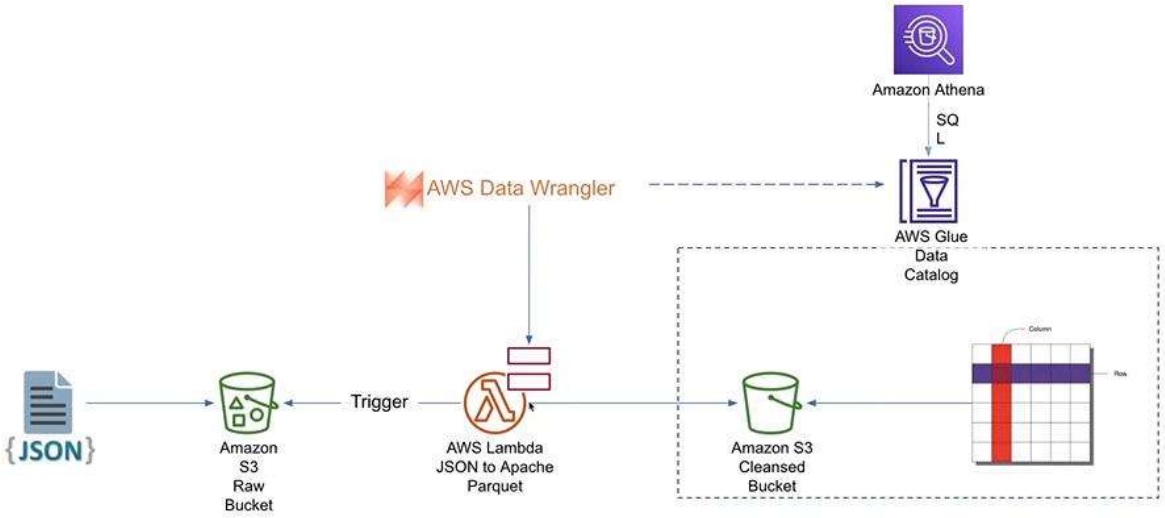
Services used : AWS Glue, AWS Lambda, Amazon S3, QuickSight, Amazon Athena, IAM.

Dataset : The data is about 211MB, in which it consists of two types of files one is json files and other is csv files. There are total 6 files in which 3 are json files and 3 are csv files.

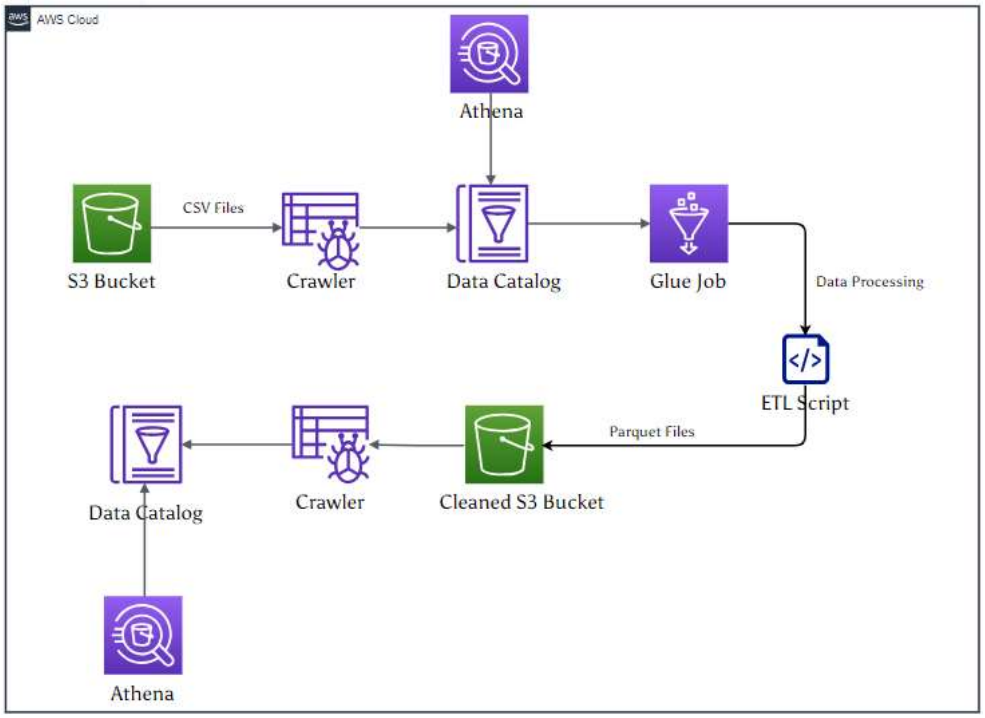
Architecture



Data Cleansing
Semi-structured data to Structured pipeline



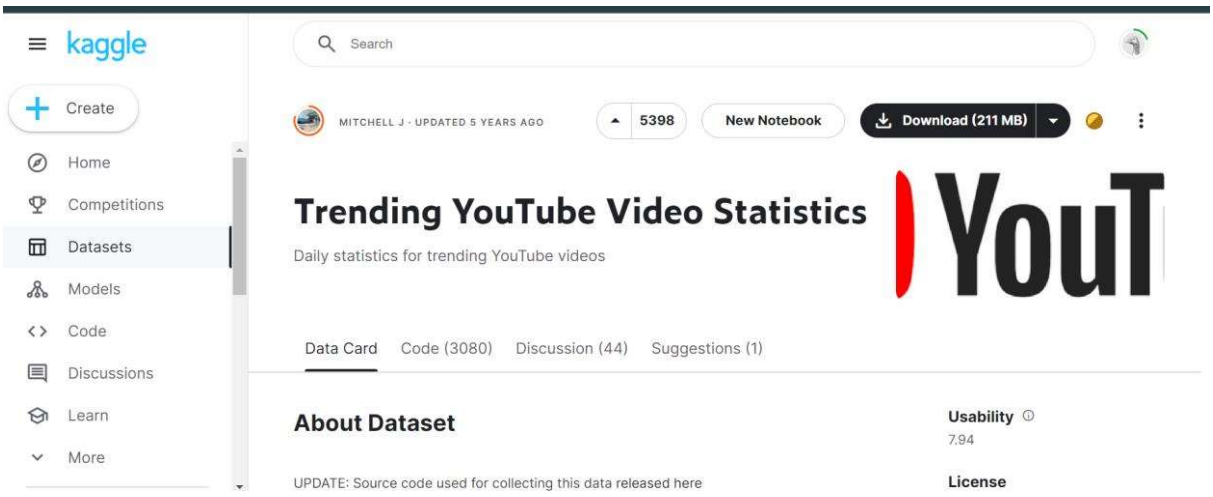
JSON Data Processing Architecture



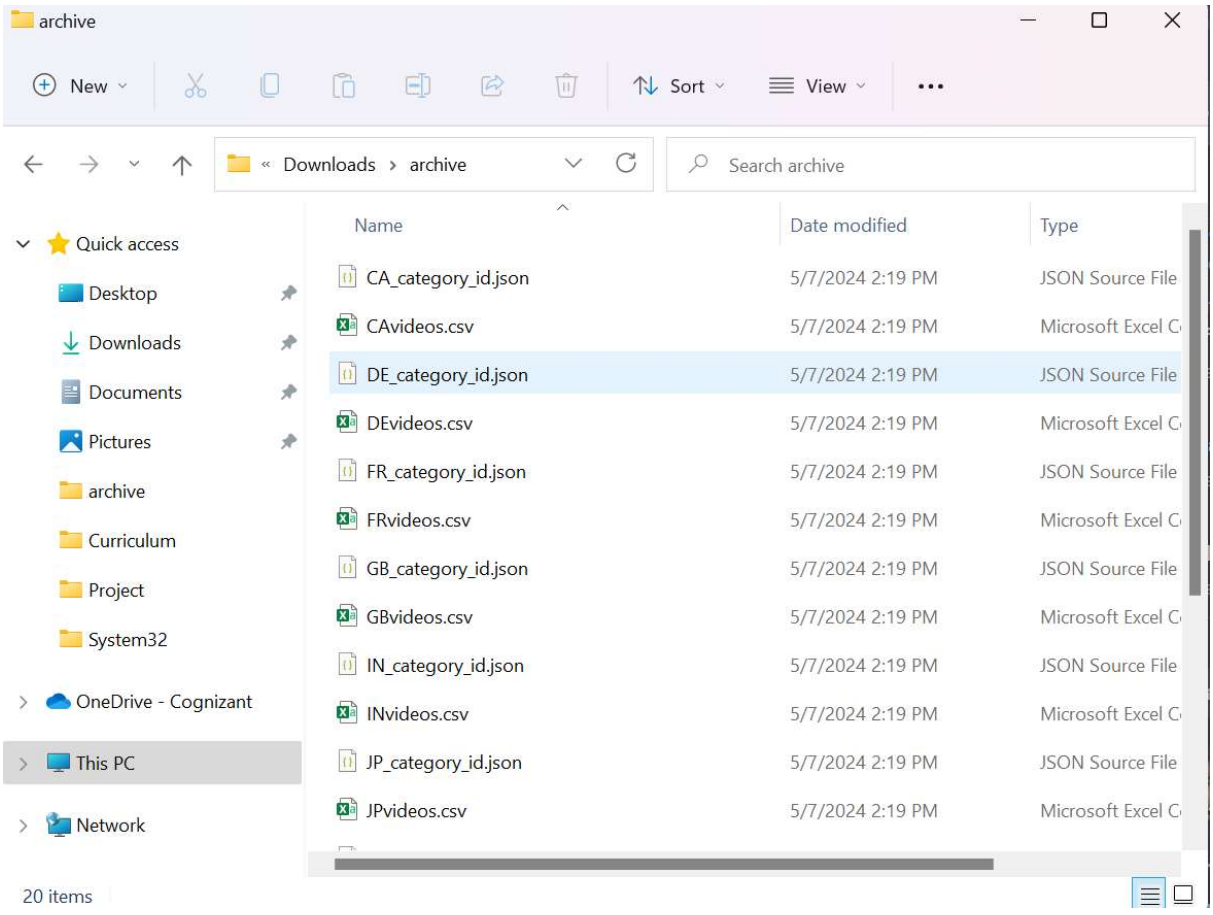
CSV Data Processing Architecture

IMPLEMENTATION

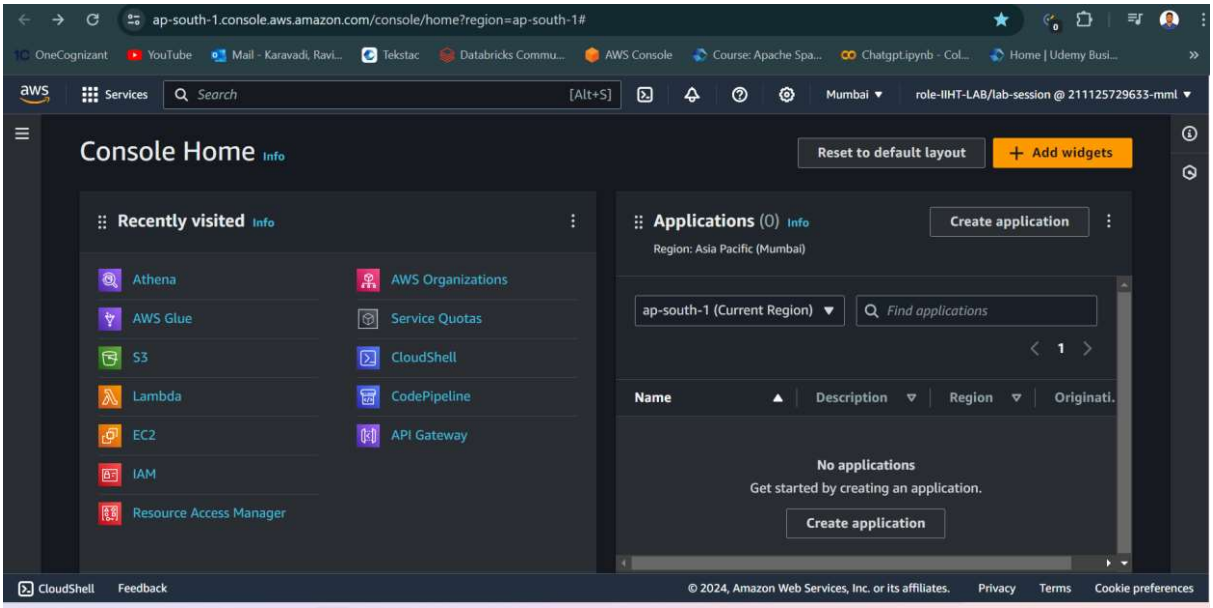
1. Download the trending Youtube data from Kaggle



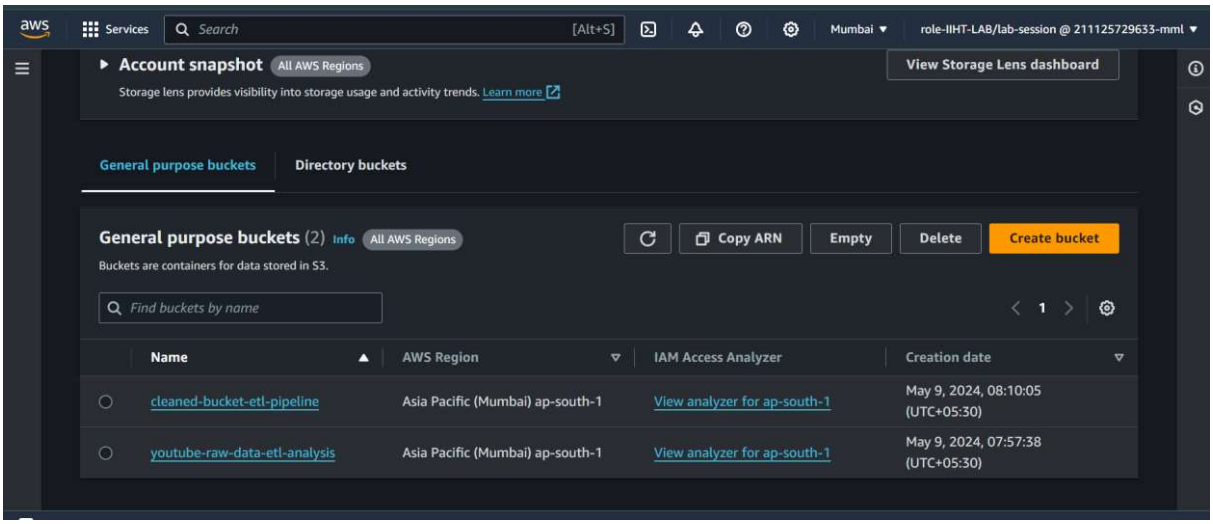
2. Now, extracted the downloaded file, It contains the data files of json and csv



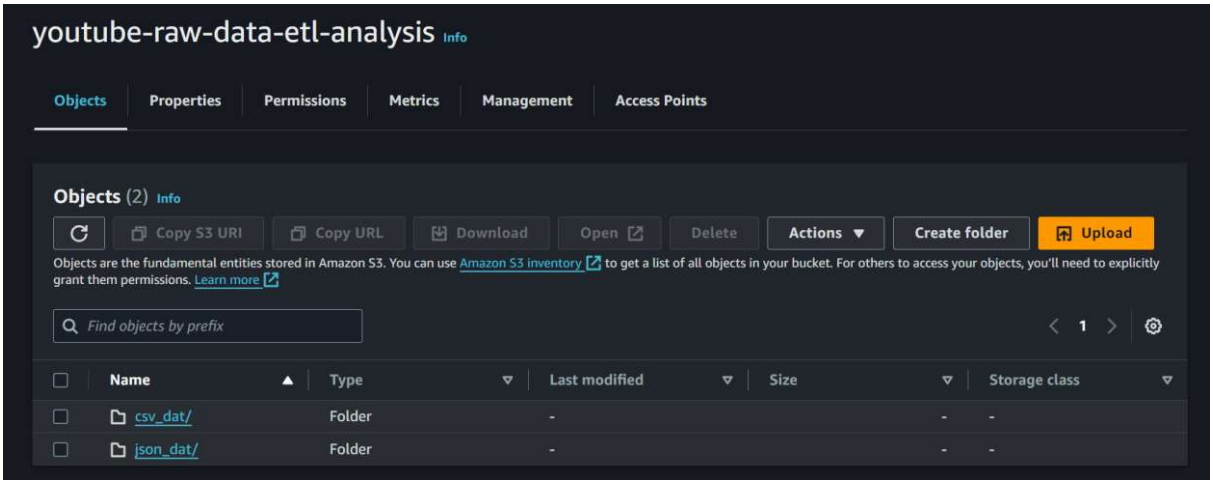
3. Login to your AWS account



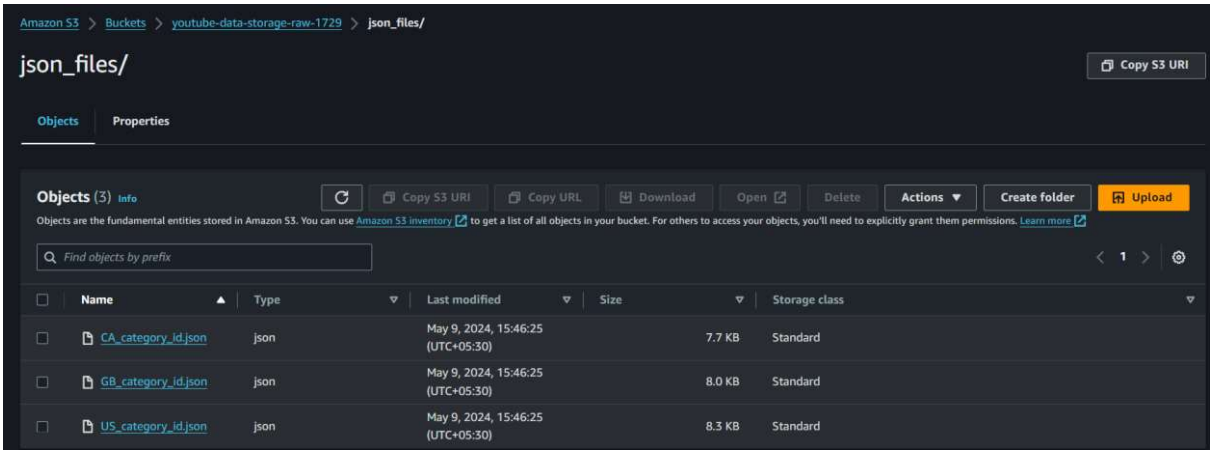
4. Navigate to S3 service, create an S3 bucket for storing the data and created another bucket for storing the cleaned data



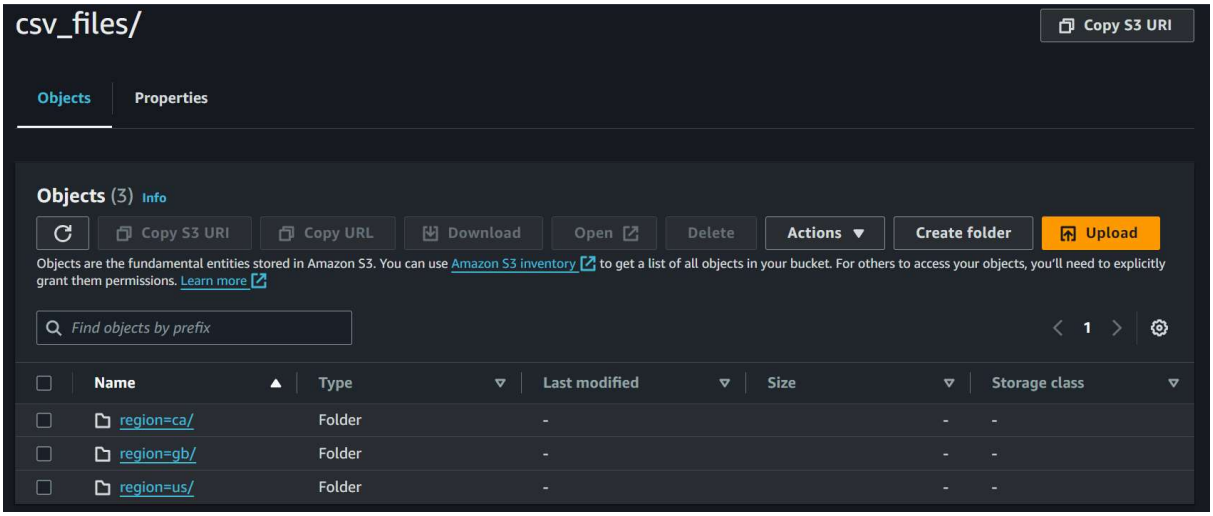
5. Now create two folders inside the S3 bucket, one for json data and one for csv files

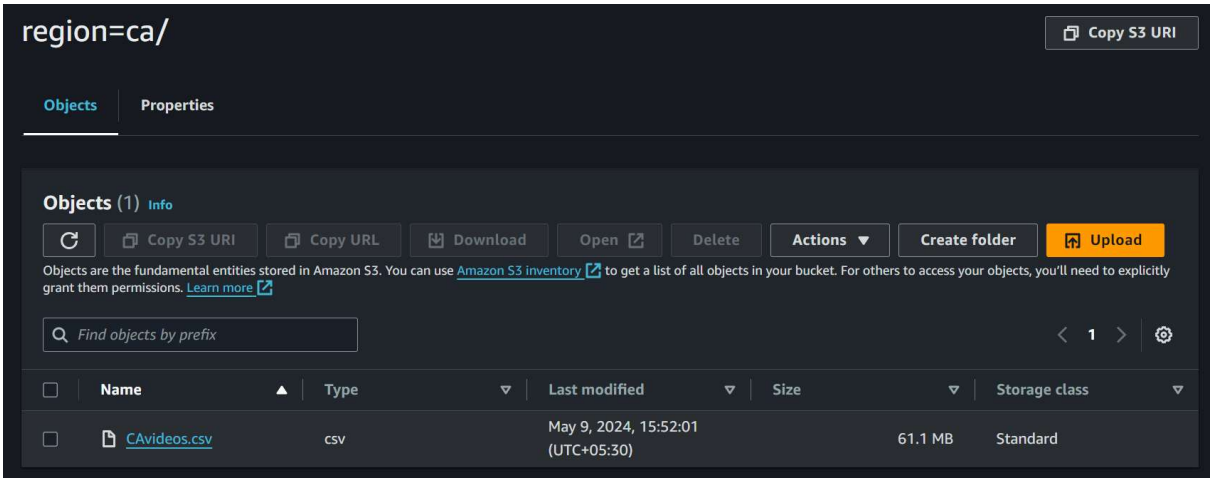


6. Navigate inside the json_dat folder and uploaded all json files present in the downloaded folder

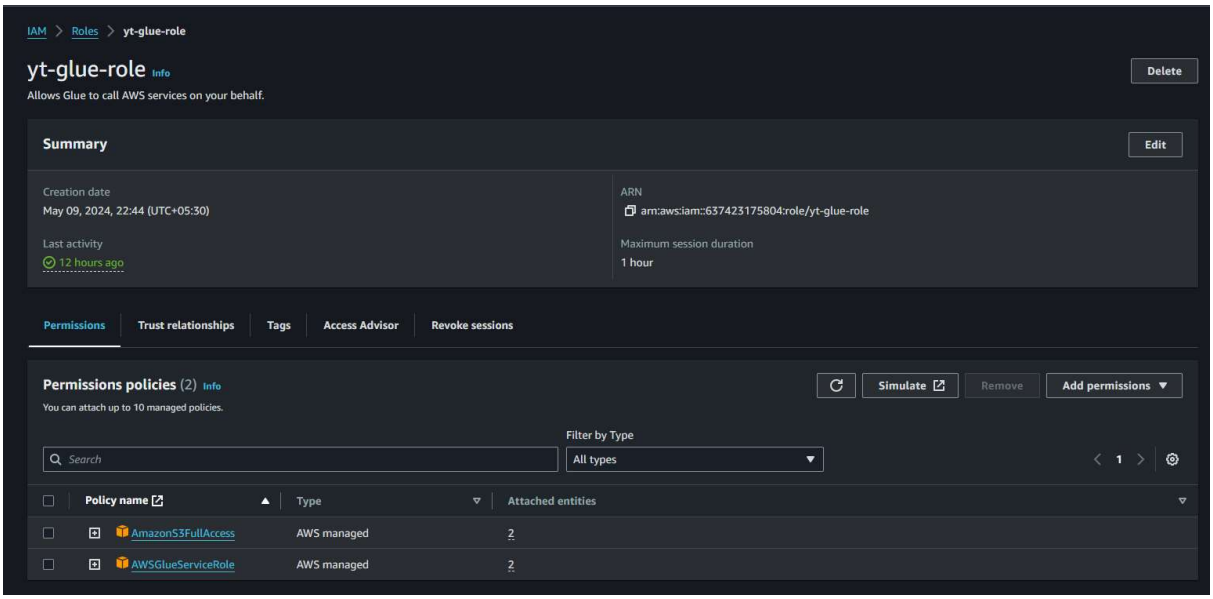


7. Now navigate to csv_dat folder and create folder for every region present in the csv_file name, and then upload the csv file specific to region, and given the folder name as region=<region_name>

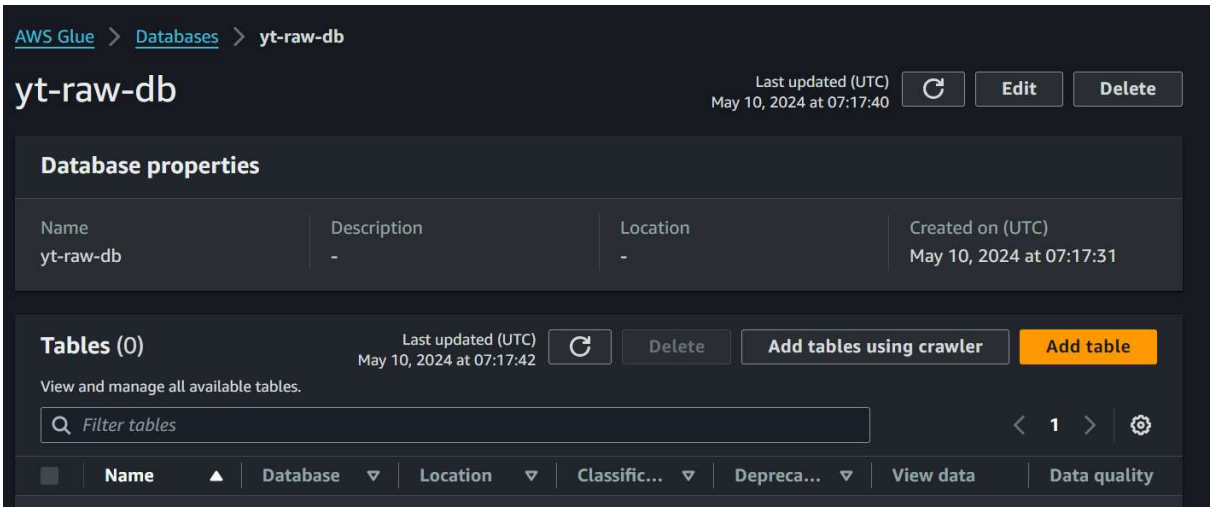




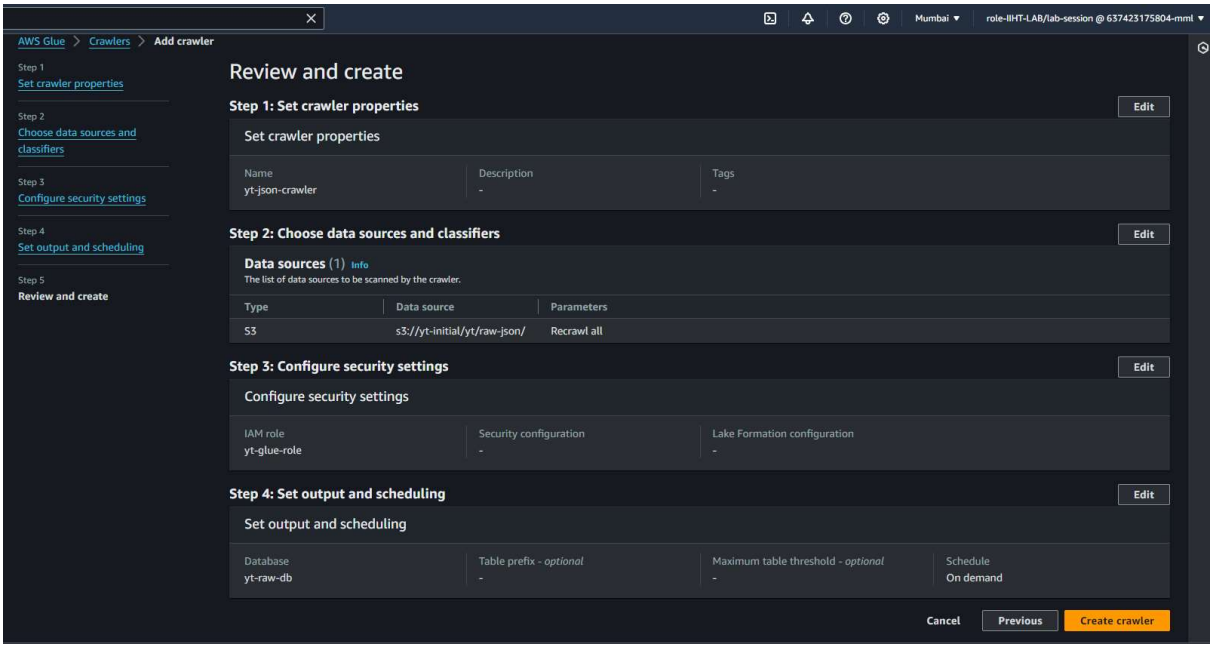
8. Create a Crawler to process JSON data
    - Crawler is glue service used to crawl data from different sources(s3) and stores the data in the glue catalog which further used for the analyical purpose.
    - Give a name to the Crawler : yt-json-crawler
    - Provide the Data Source loation of json files for the crawler : s3location
    - Create IAM role for the Crawler with necessary policies(S3FullAccess, GlueService) and assign the role to the Crawler



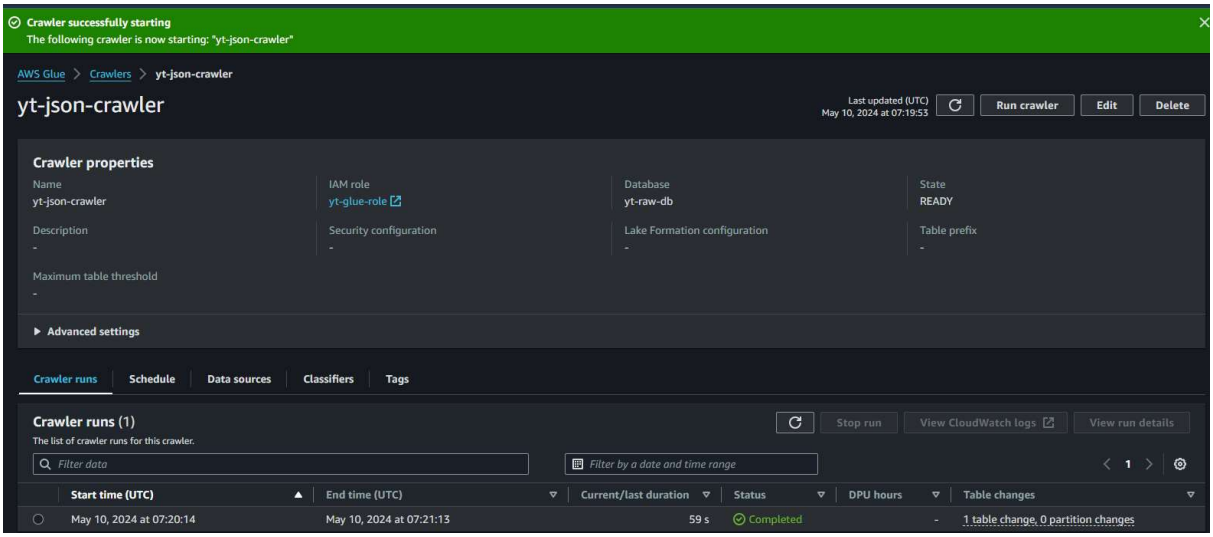
- Create a Database for the Crawler to create Table and store the data, In short the Data Catalog.



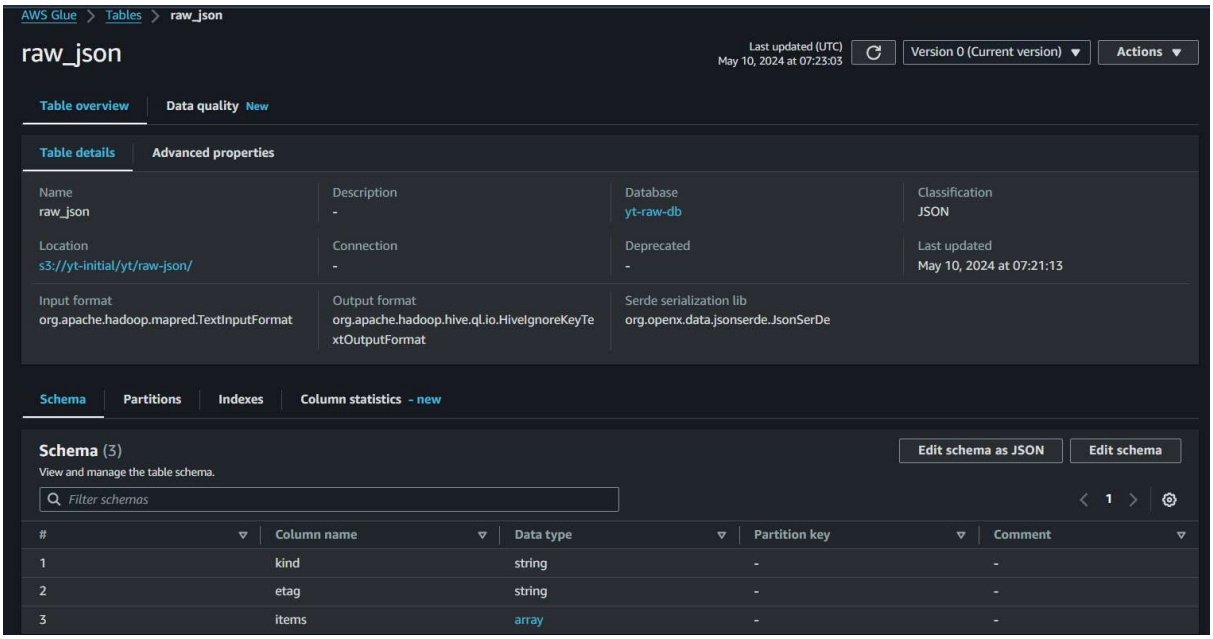
- Review and Create the Crewler



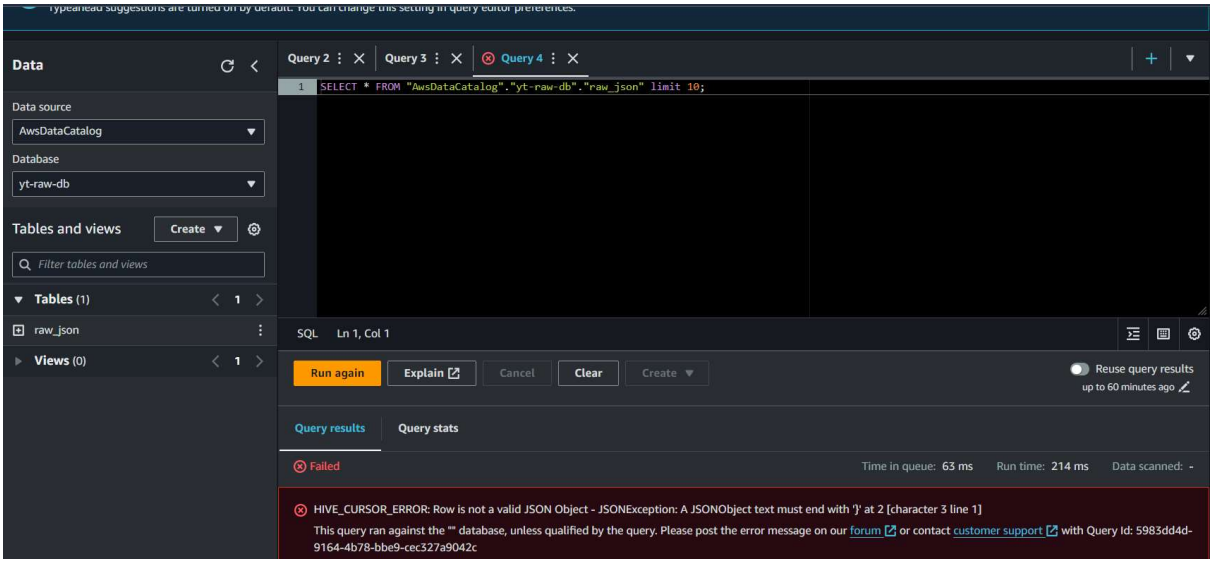
9. Run the Crawler
    - Run the Crawler and wait for the data catalog to be formed i.e. the Table with the crawled data.
    - There create a table named raw_json



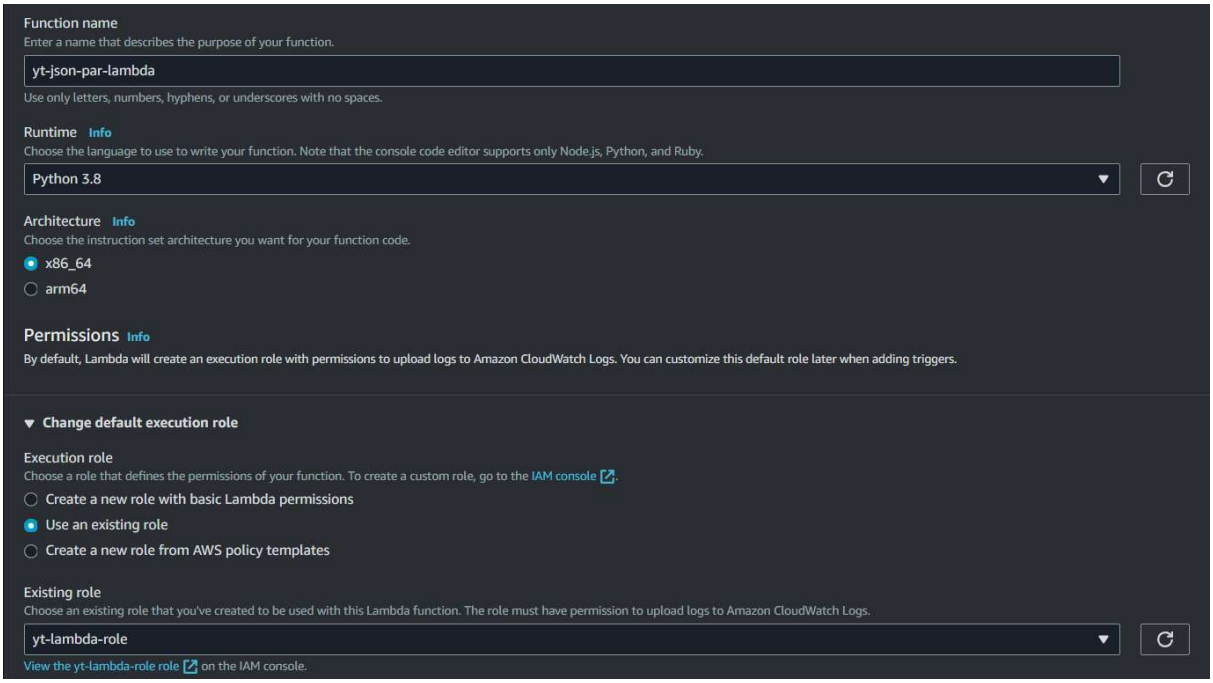
10. Review the Table and Perform the Query analysis
    - The Crawler will create a table with the json data in the specified database location



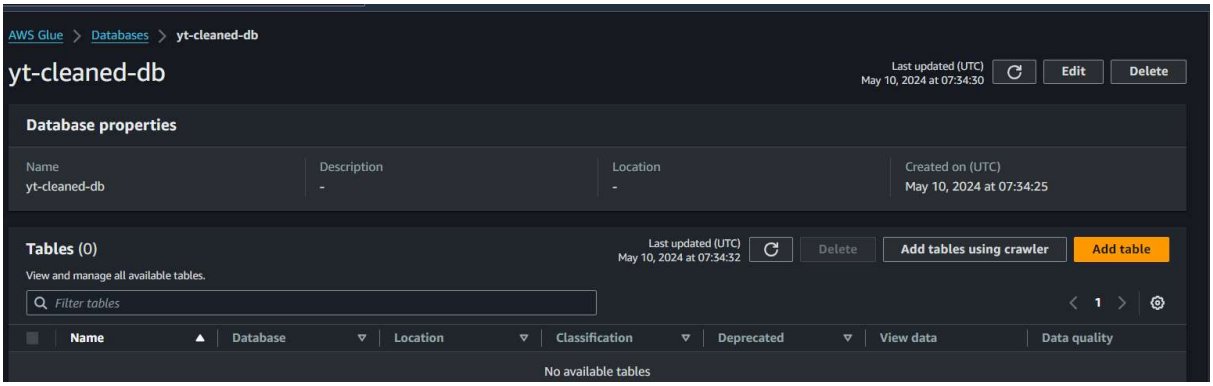
- View the data in the table through the Amazon Athena and Run the Query for the retrieval of the data.
- Here comes an error message(JSON Exception) stating that improper json format, so that unable to fetch the data while Querying.
- To Overcome the above Error. We create a Lambda for extratction of specified data.



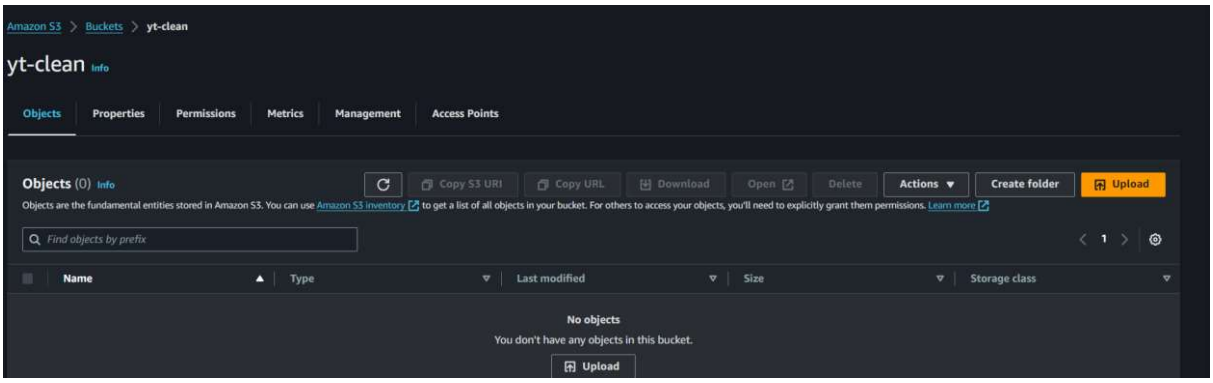
11. Lambda Function Creation
    - We create the lambda function for the extraction of only desired data from the json file that convert the json to parquet format which can be readability for the glue and Lambda
    - Give a name to Lambda Function : yt-json-par-lambda
    - Select the runtime Engine : Python 3.8
    - Create and attach the IAM Role for the Lambda with necessary permissions and Policies(S3FullAccess, GlueService)



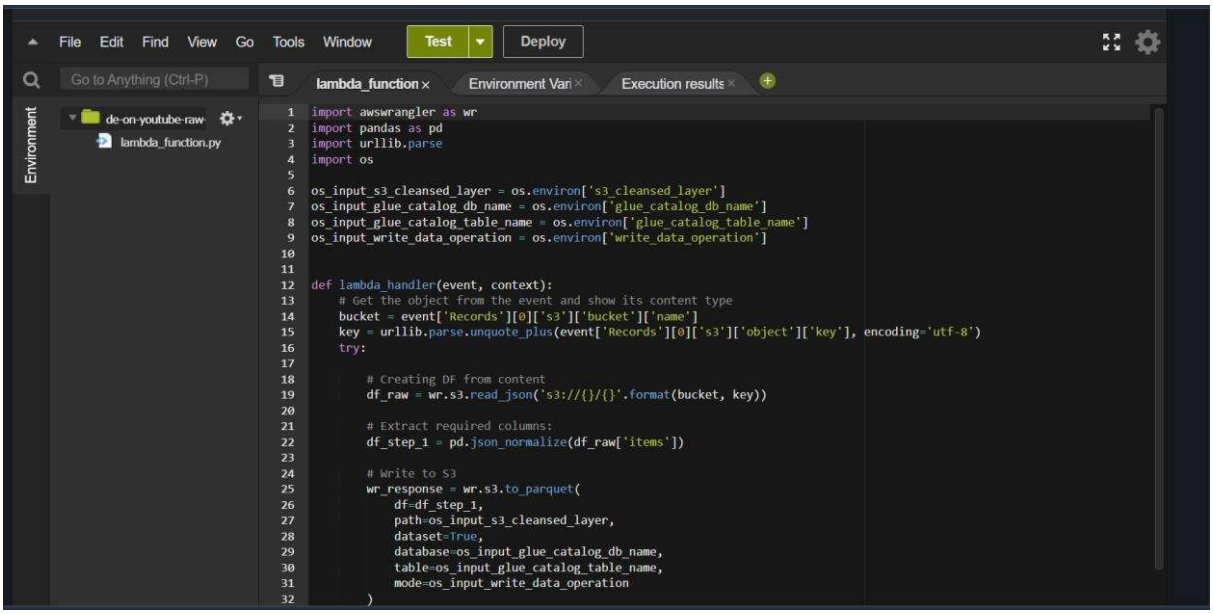
- Create a new Database(yt-cleane-db) for the storage of the cleaned json table for analytical purposes



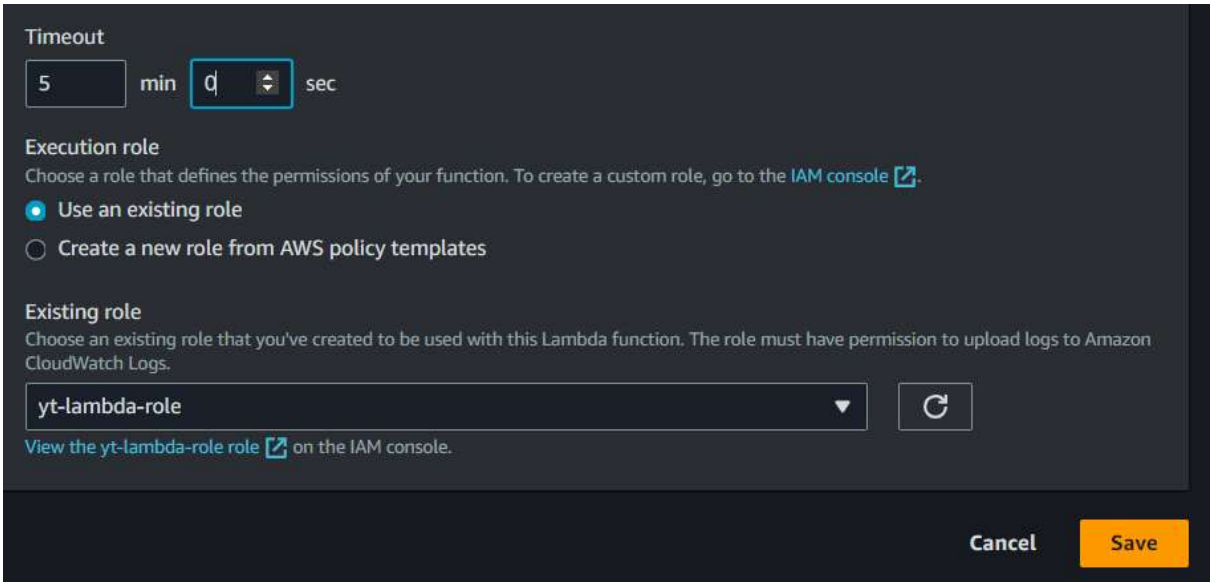
- Create a New S3 Bucket for the storage of the all cleaned data. here in this case all cleaned data parquets of both csv and json file will be stored



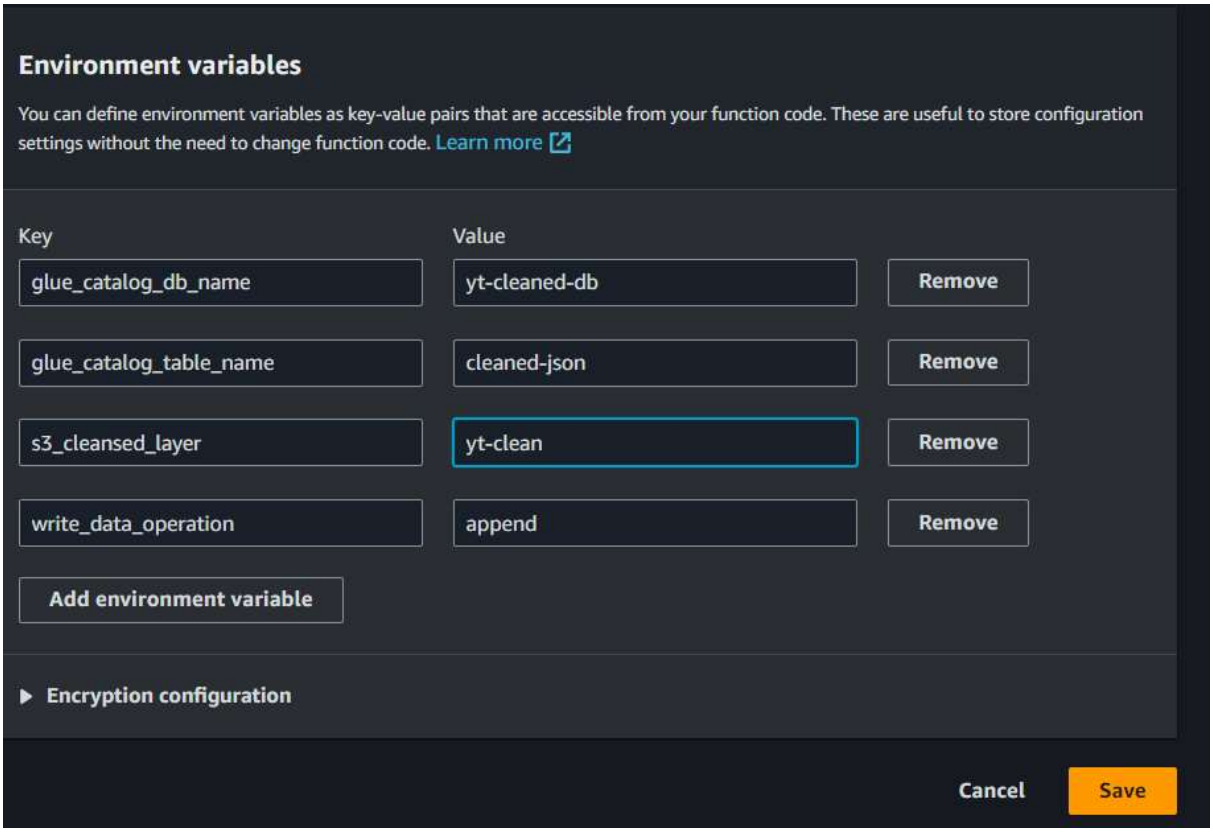
- Customise the lambda function, so that it extracts only needed data from the json file and the store it in a new file format Parquet and stores these parquets in new bucket and Also a table is created in a database.



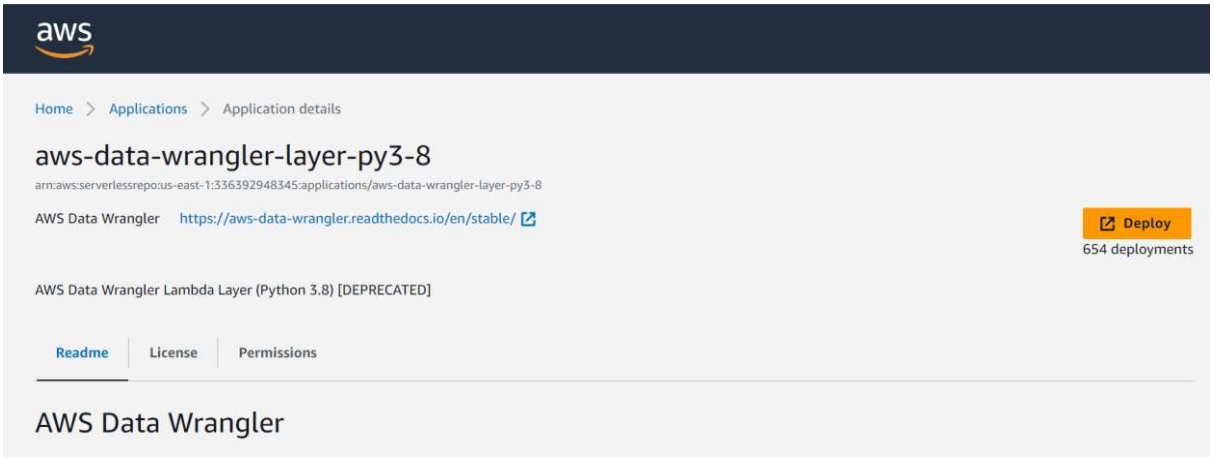
- Configure the default timeout time and set it to 5 mins in the General Configuration tab

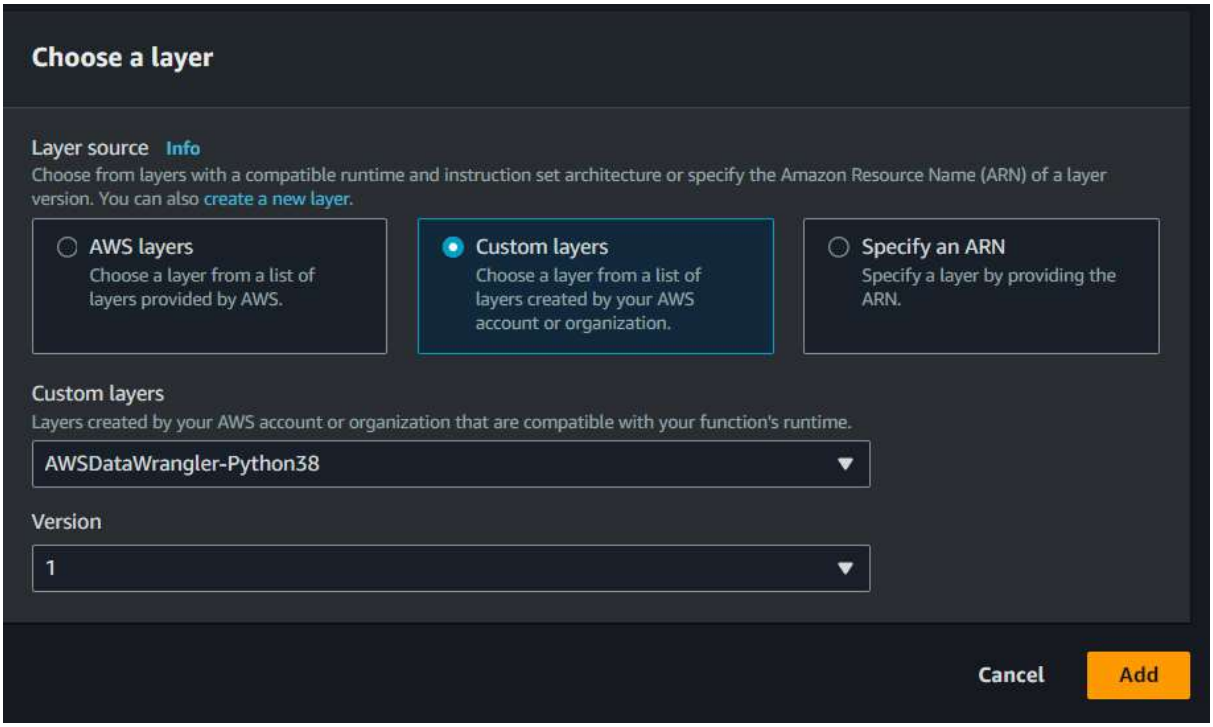


- Configure the Environment variables such S3 target location, Database location, Table name, Write operation the lambda.

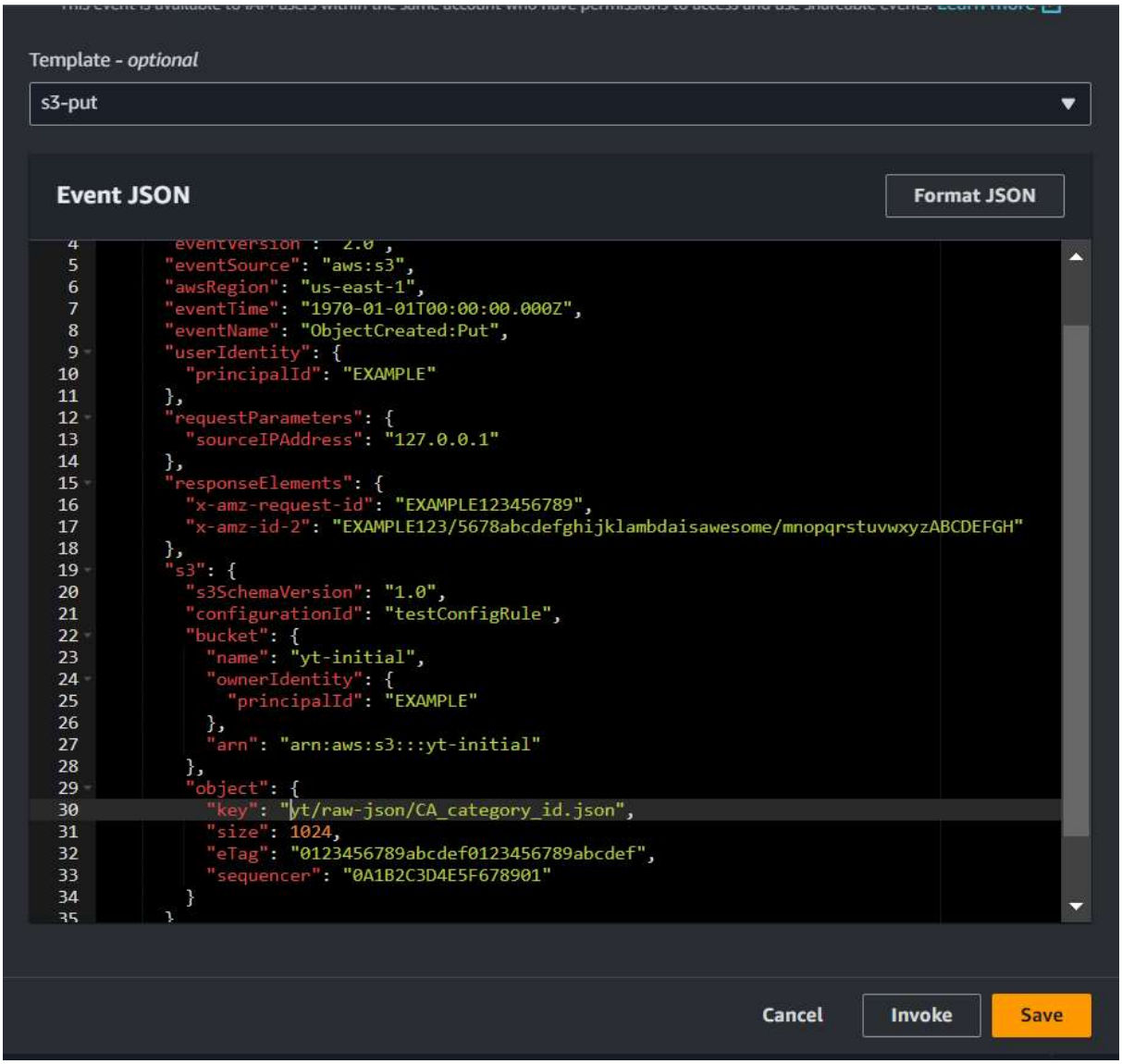


- Add AWSDataWrangler Layer which helps to provide compute environment for the lambda executions uninterruptedly.

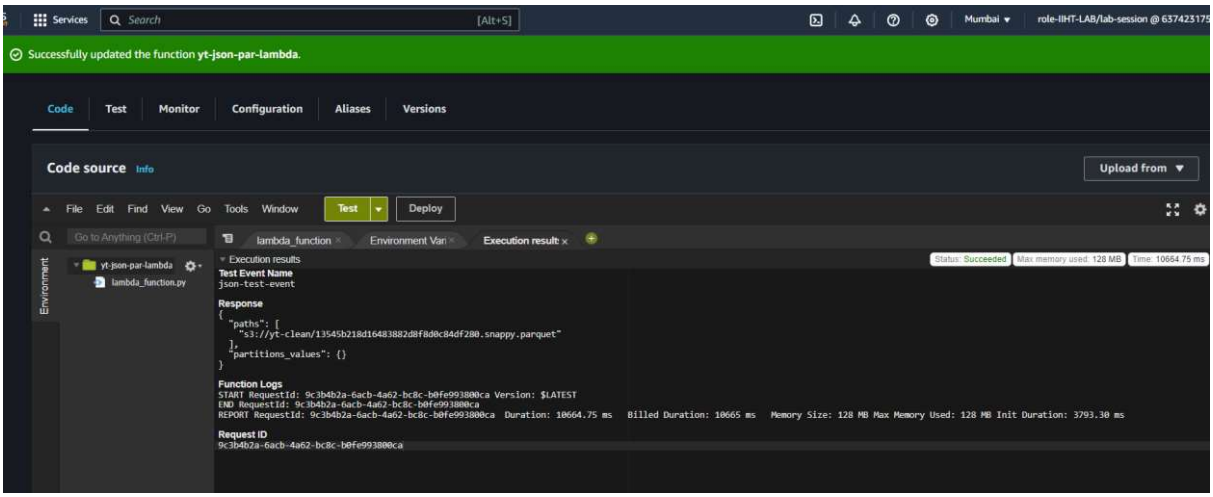




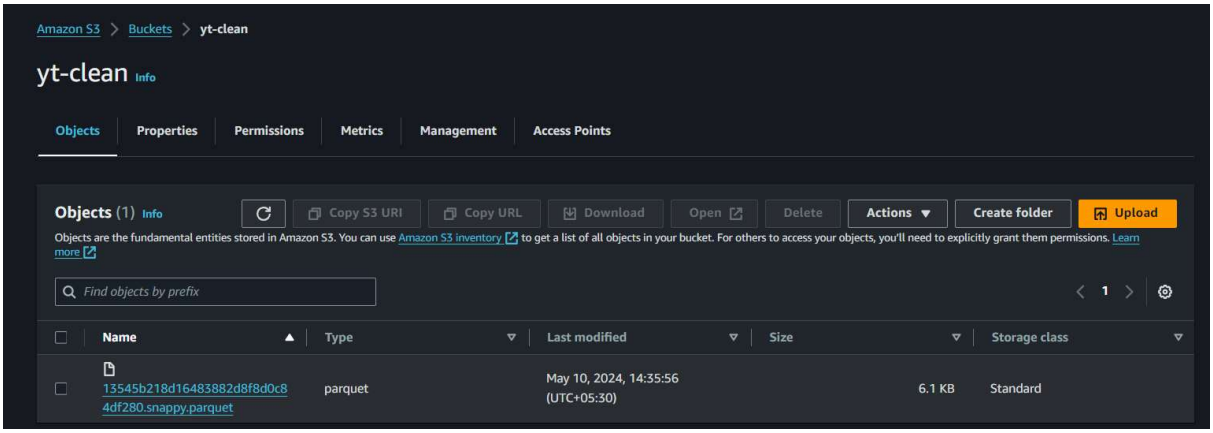
- Create a Test event for Deploying and Testing the code that we have written, in the scrip of the event provide some details like (template : S3PUT) and source bucket name and the key of the files stored in the bucket



- Test the Lambda Function and Check for the Results
- It should create the parquet files in the bucket and the Tables in the database.

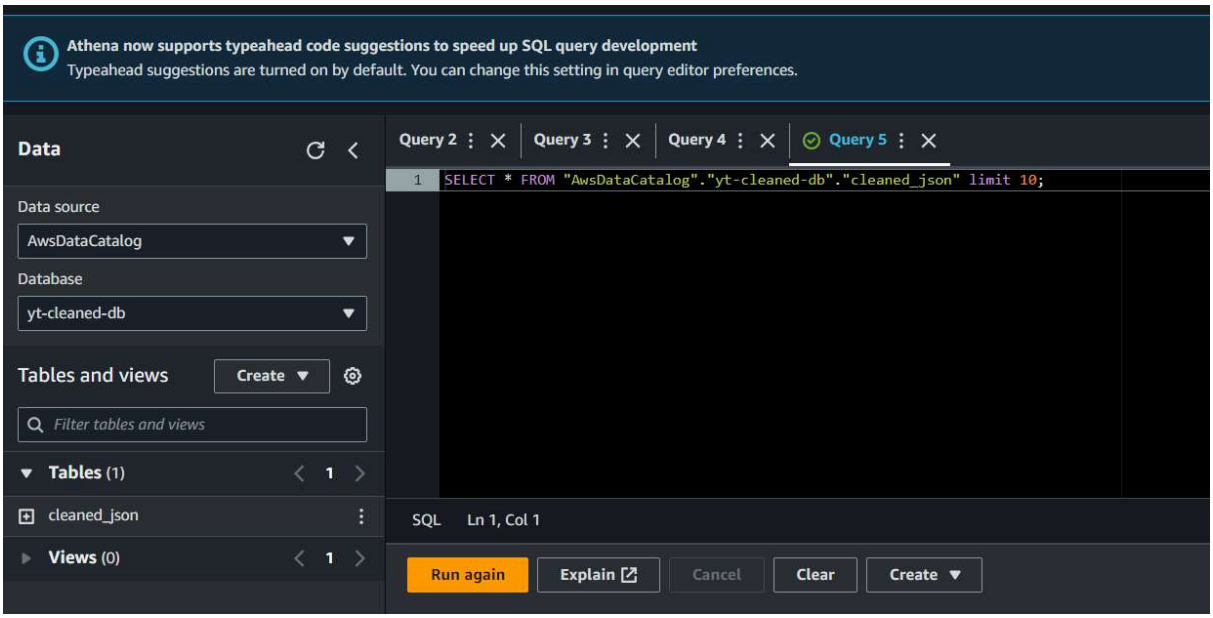


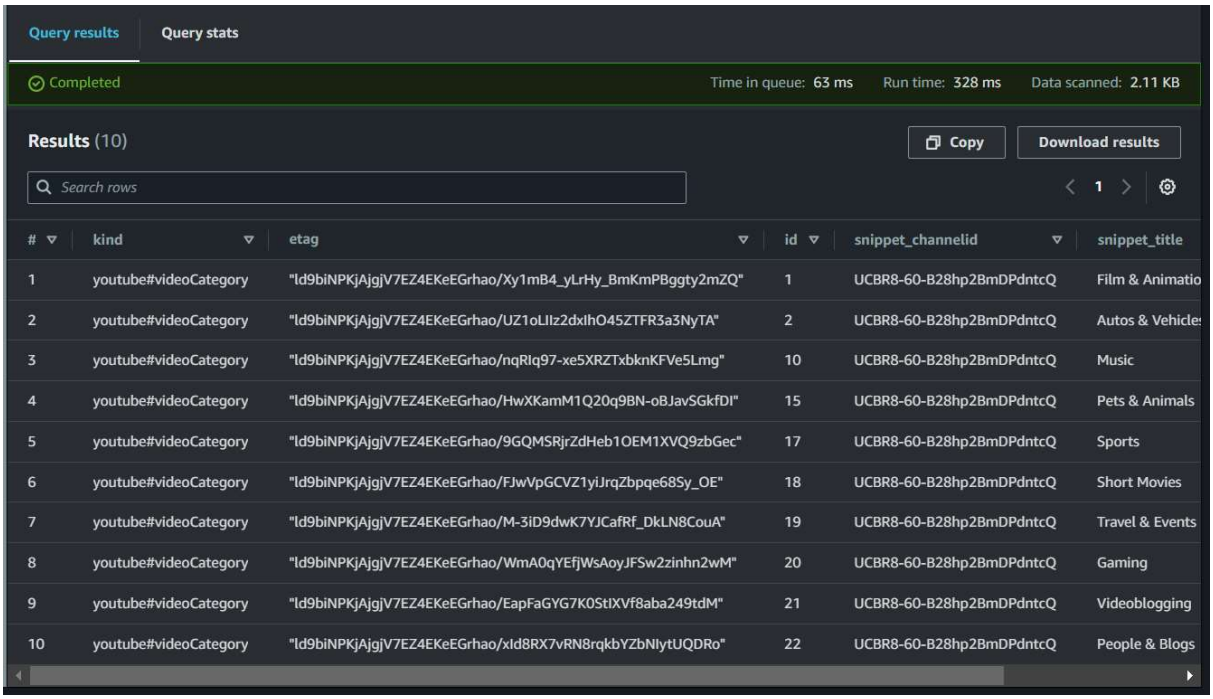
- Check for the Confirmation of Parquet file generation



- Check whether the table is created or not in the cleaned database

- View the table data through the Athena Query Editor and perform a simply query and check for the proper execution.
- Here, It proves that the Error we encountered earlier was rectified and can proceed with this new data.





Query results | Query stats

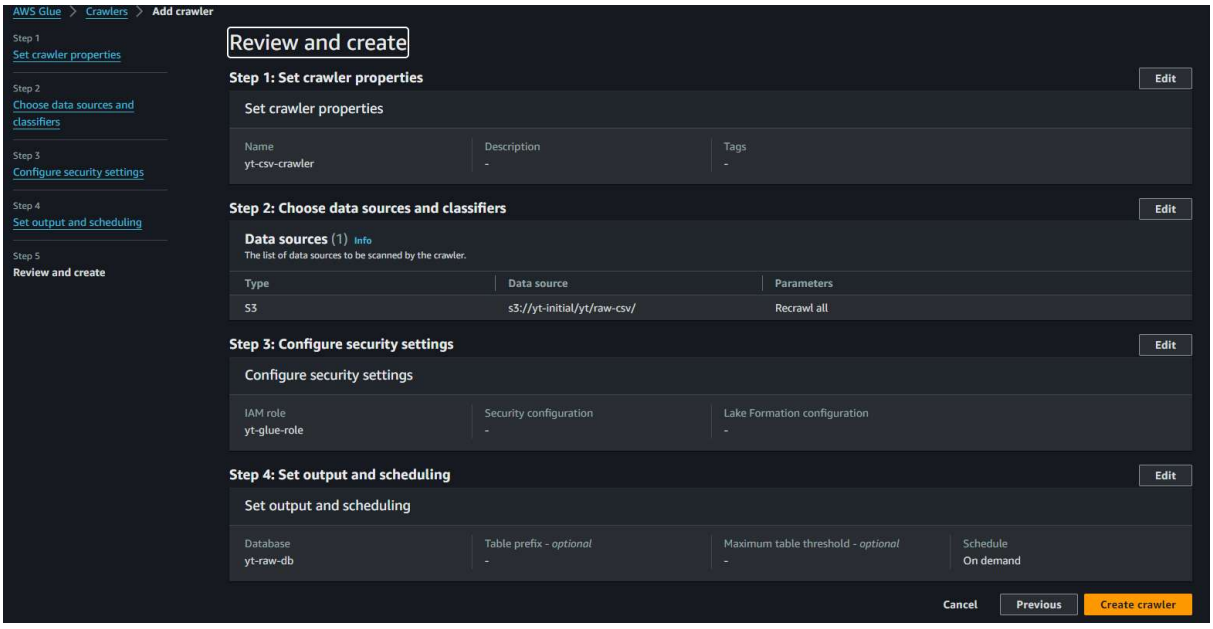
Completed — Time in queue: 63 ms — Run time: 328 ms — Data scanned: 2.11 KB

Results (10) — Copy — Download results

Search rows — 1

#	kind	etag	id	snippet_channelid	snippet_title
1	youtube#videoCategory	"ld9biNPKjAjgjV7EZ4EKeEGrhao/Xy1mB4_yLrHy_BmKmPBggty2mZQ"	1	UCBR8-60-B28hp2BmDPdntcQ	Film & Animatio
2	youtube#videoCategory	"ld9biNPKjAjgjV7EZ4EKeEGrhao/UZ1oLIIz2dxlhO45ZTFR3a3NyTA"	2	UCBR8-60-B28hp2BmDPdntcQ	Autos & Vehicles
3	youtube#videoCategory	"ld9biNPKjAjgjV7EZ4EKeEGrhao/nqRIq97-xe5XRZTxbknKFVe5Lmg"	10	UCBR8-60-B28hp2BmDPdntcQ	Music
4	youtube#videoCategory	"ld9biNPKjAjgjV7EZ4EKeEGrhao/HwXKamM1Q20q9BN-oBJavSGkfDI"	15	UCBR8-60-B28hp2BmDPdntcQ	Pets & Animals
5	youtube#videoCategory	"ld9biNPKjAjgjV7EZ4EKeEGrhao/9GQMSRjrZdHeb1OEM1XVQ9zbGec"	17	UCBR8-60-B28hp2BmDPdntcQ	Sports
6	youtube#videoCategory	"ld9biNPKjAjgjV7EZ4EKeEGrhao/FJwVpGCVZ1yiJrqZbpqe68Sy_OE"	18	UCBR8-60-B28hp2BmDPdntcQ	Short Movies
7	youtube#videoCategory	"ld9biNPKjAjgjV7EZ4EKeEGrhao/M-3iD9dwK7YJCafRf_DkLN8CouA"	19	UCBR8-60-B28hp2BmDPdntcQ	Travel & Events
8	youtube#videoCategory	"ld9biNPKjAjgjV7EZ4EKeEGrhao/WmA0qYEfjWsAoyJFSw2zinhn2wM"	20	UCBR8-60-B28hp2BmDPdntcQ	Gaming
9	youtube#videoCategory	"ld9biNPKjAjgjV7EZ4EKeEGrhao/EapFaGYG7K0StIXVf8aba249tdM"	21	UCBR8-60-B28hp2BmDPdntcQ	Videoblogging
10	youtube#videoCategory	"ld9biNPKjAjgjV7EZ4EKeEGrhao/xId8RX7vRN8rqkbYZbNIytUQDRo"	22	UCBR8-60-B28hp2BmDPdntcQ	People & Blogs

12. Create a Crawler to process csv Files
    - Configure the crawler with the details like, Set the crawler properties, Data source locations, Security Settings, Output location and scheduling.



AWS Glue > Crawlers > Add crawler

Step 1 Set crawler properties
Step 2 Choose data sources and classifiers
Step 3 Configure security settings
Step 4 Set output and scheduling
Step 5 Review and create

Review and create

Step 1: Set crawler properties — Edit

Set crawler properties

Name	Description	Tags
yt-csv-crawler	-	-

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://yt-initial/yt/raw-csv/	Recrawl all

Step 3: Configure security settings — Edit

Configure security settings

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

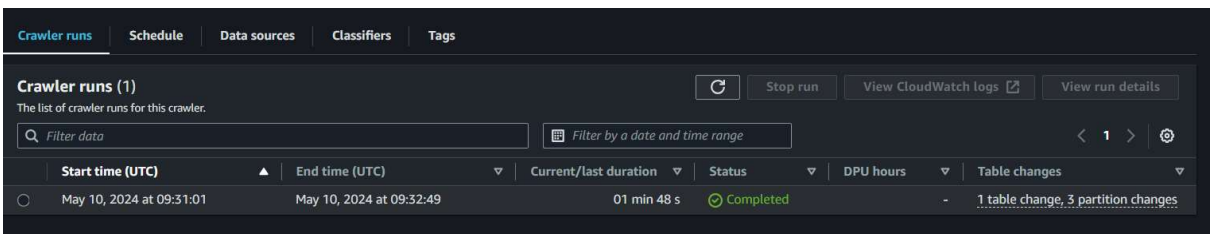
Step 4: Set output and scheduling — Edit

Set output and scheduling

Database	Table prefix - *optional*	Maximum table threshold - *optional*	Schedule
yt-raw-db	-	-	On demand

Cancel — Previous — Create crawler

13. Run the Crawler
    - Same as previous, It creates data catalog table.



Crawler runs | Schedule | Data sources | Classifiers | Tags

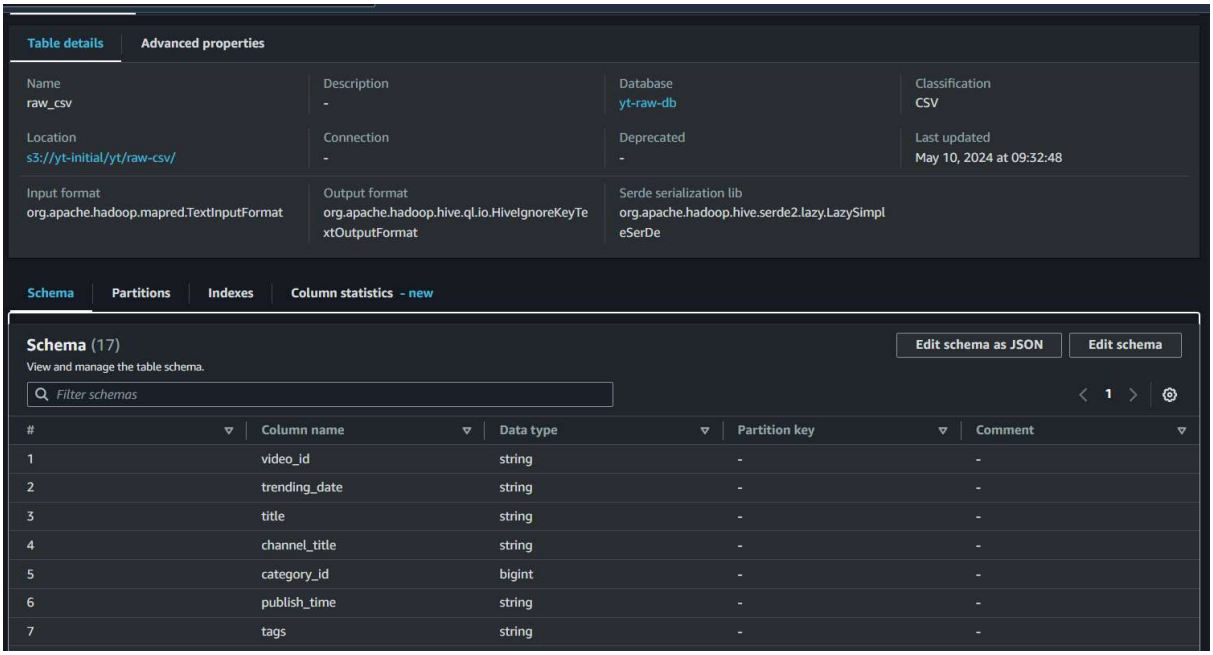
Crawler runs (1)
The list of crawler runs for this crawler.

Stop run — View CloudWatch logs — View run details

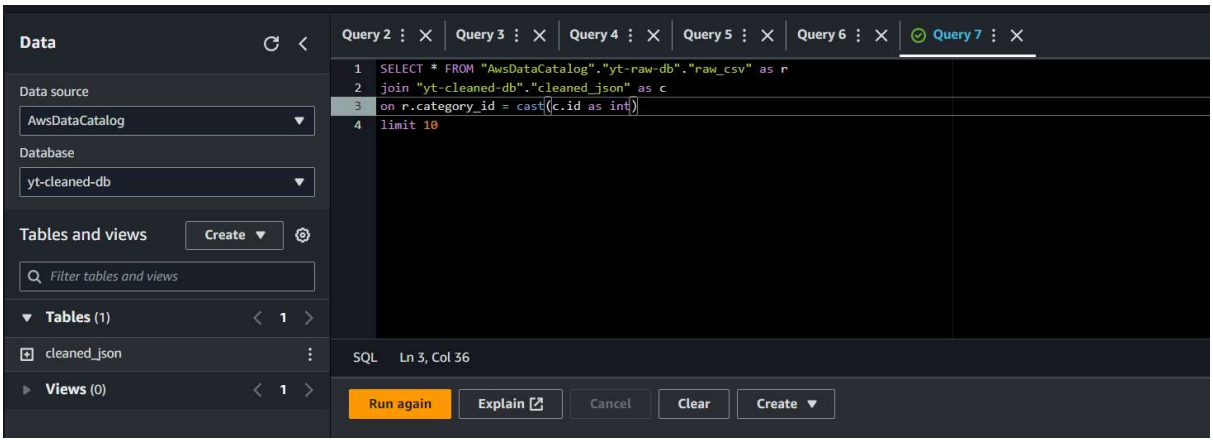
Filter data — Filter by a date and time range — 1

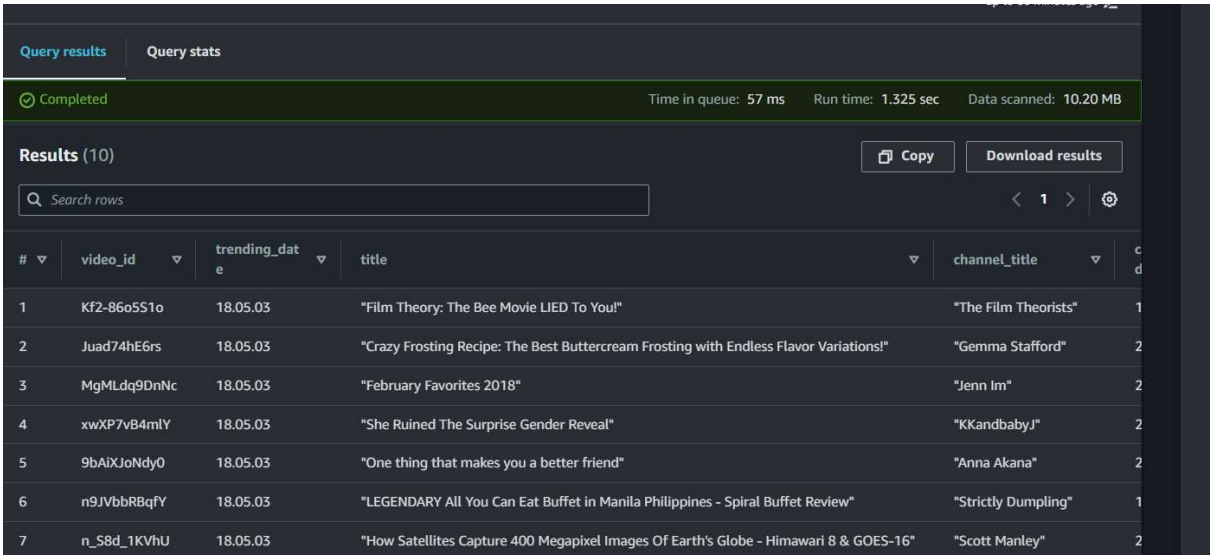
Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
May 10, 2024 at 09:31:01	May 10, 2024 at 09:32:49	01 min 48 s	Completed	-	1 table change, 3 partition changes

- Here it creates a partitioned table with region as partition key.

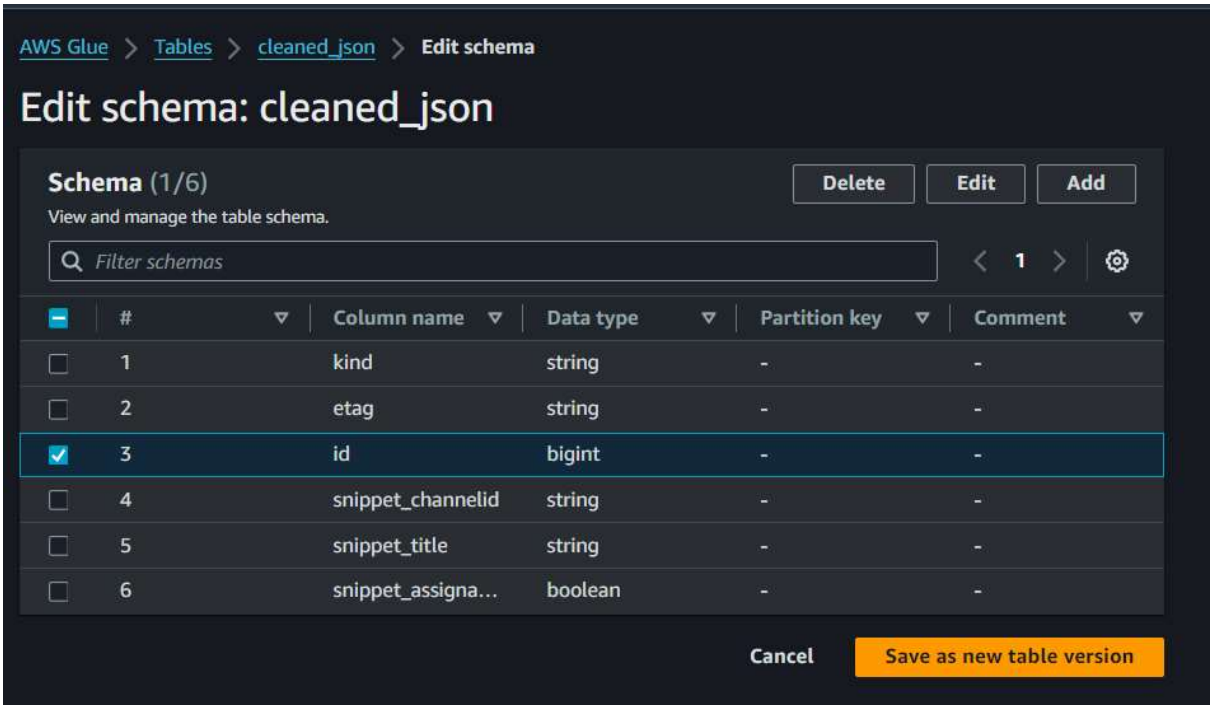


14. View the Table data and Perform some Query operations through Athena.
    - Try joining the two table(json, csv) on a joining condition and query the data, But here we need TypeCast datatype of column in joining condition, inorder to overcome the mismatch error.

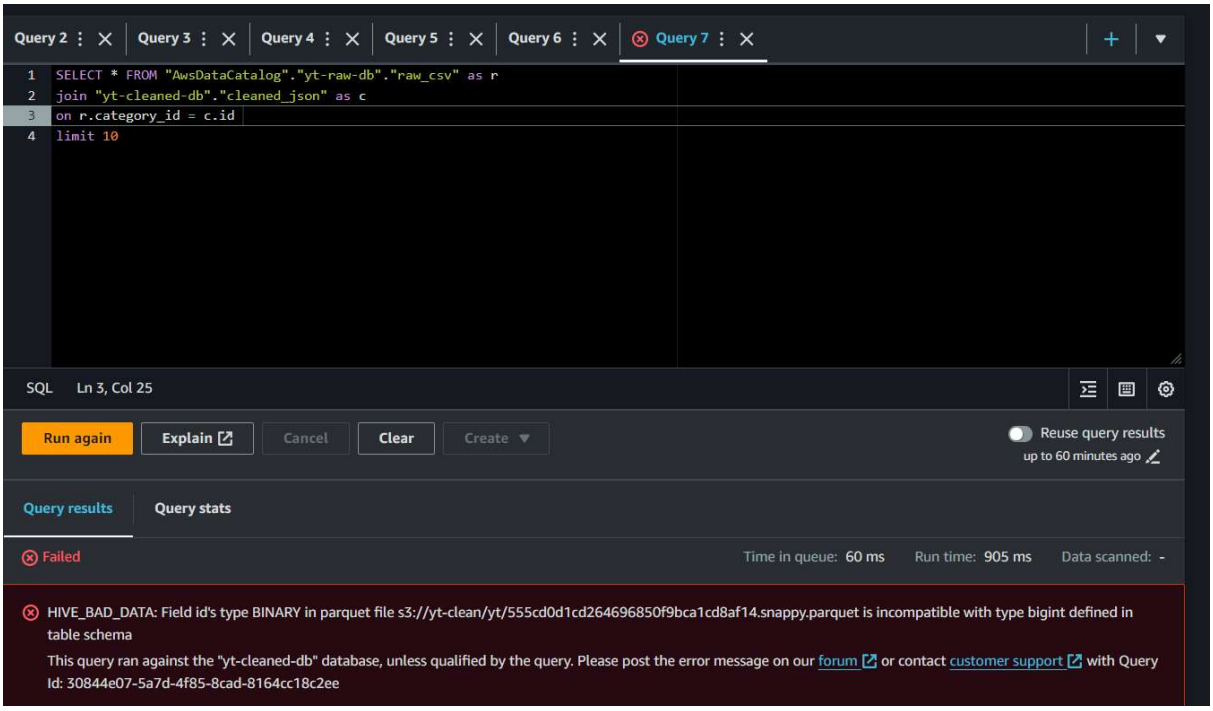




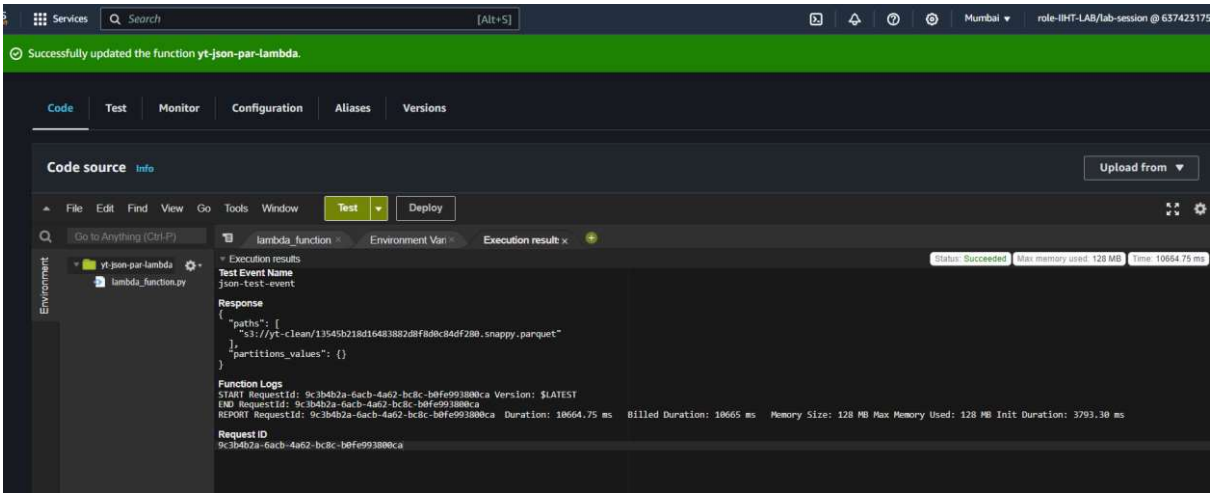
- But, Typecasting the data is not a good practice, So we change the schema of the table Instead.



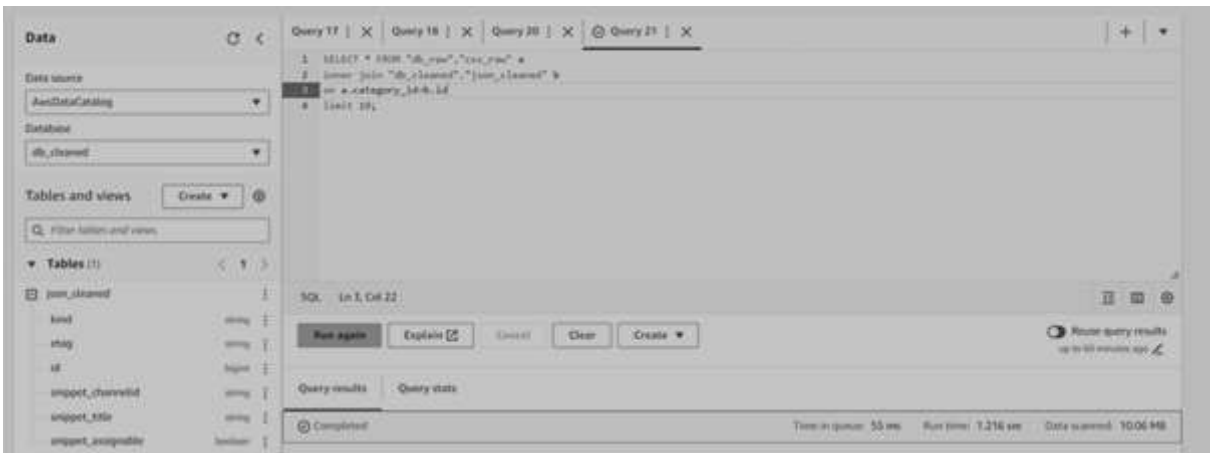
- Now after changing the schema and try to query data, Even then we encounter an Error message(Hive Data Error)
- This means changing schema on local table itself does not reflect in the parquet file that is generated previously.



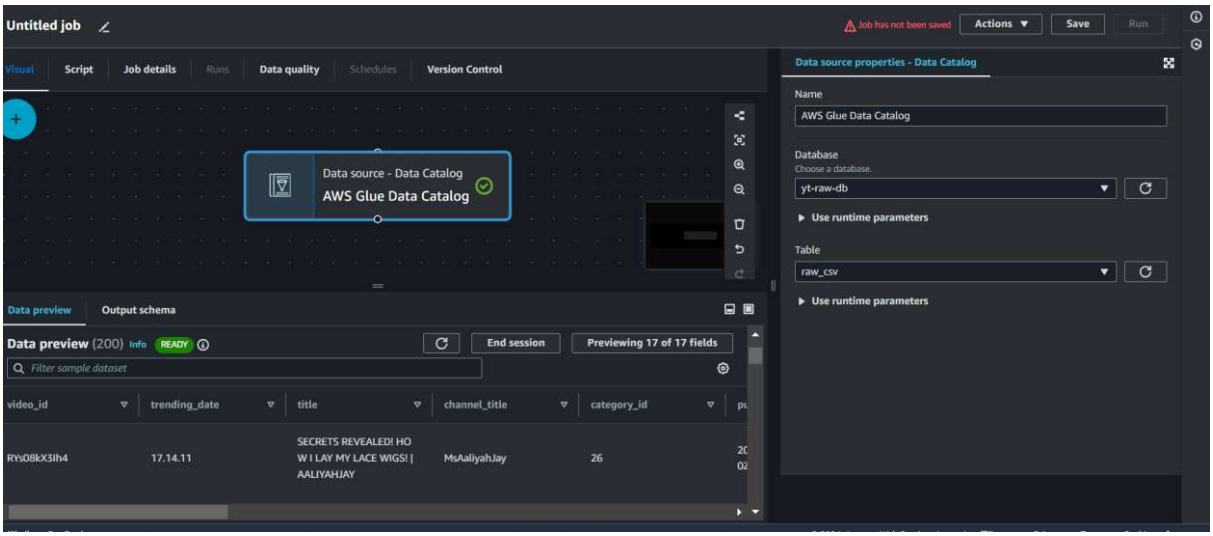
- To overcome this, we need to delete the previously generated parquet file and rerun the lambda to generate a new parquet with all changes in the schema reflected in it.



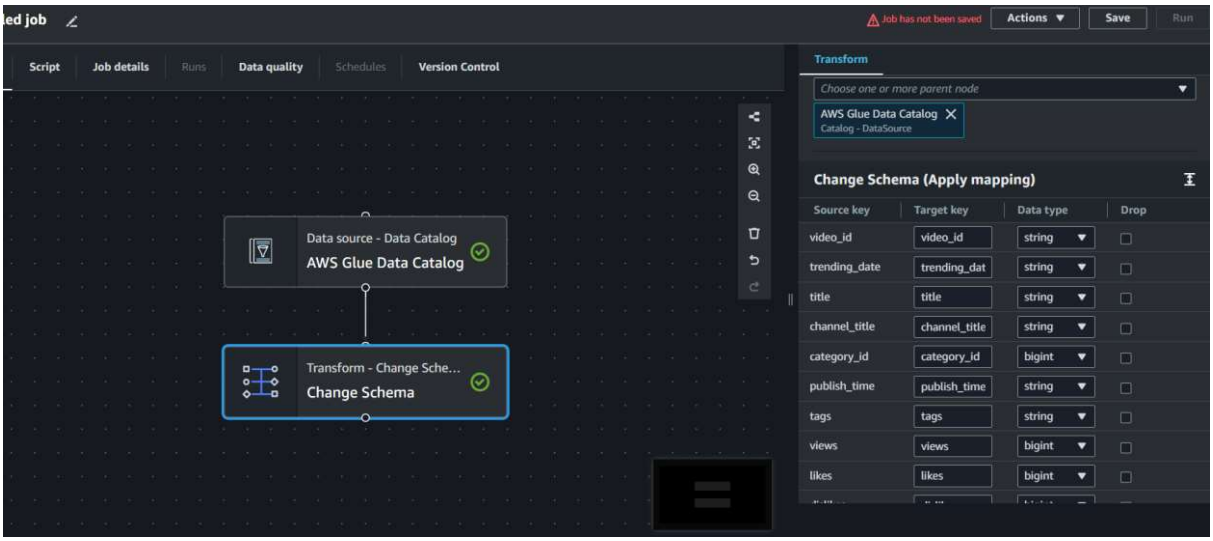
- Now, Run the Query again and we see that there are no errors while running it.



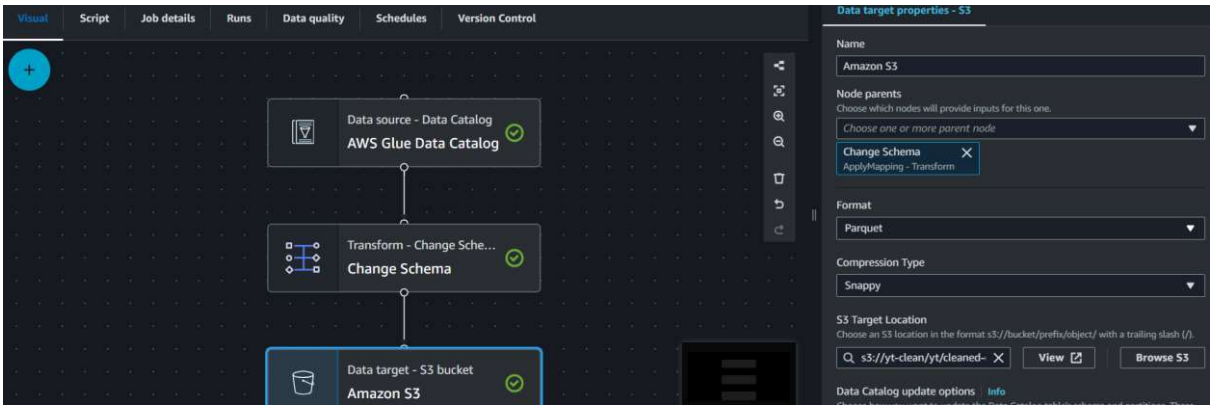
15. Create a ETL job for the csv to parquet conversion
    - Create a Glue ETL job that converts the raw csv files into new cleansed data format i.e. into the parquet format. We use Visual ETL tool for the designing the job and provide all the necessary details.
    - Select the Data source, in this case, AWS Glue Data Catalog
    - Choose the database and raw csv table
    - Provide the IAM role for the Glue Job

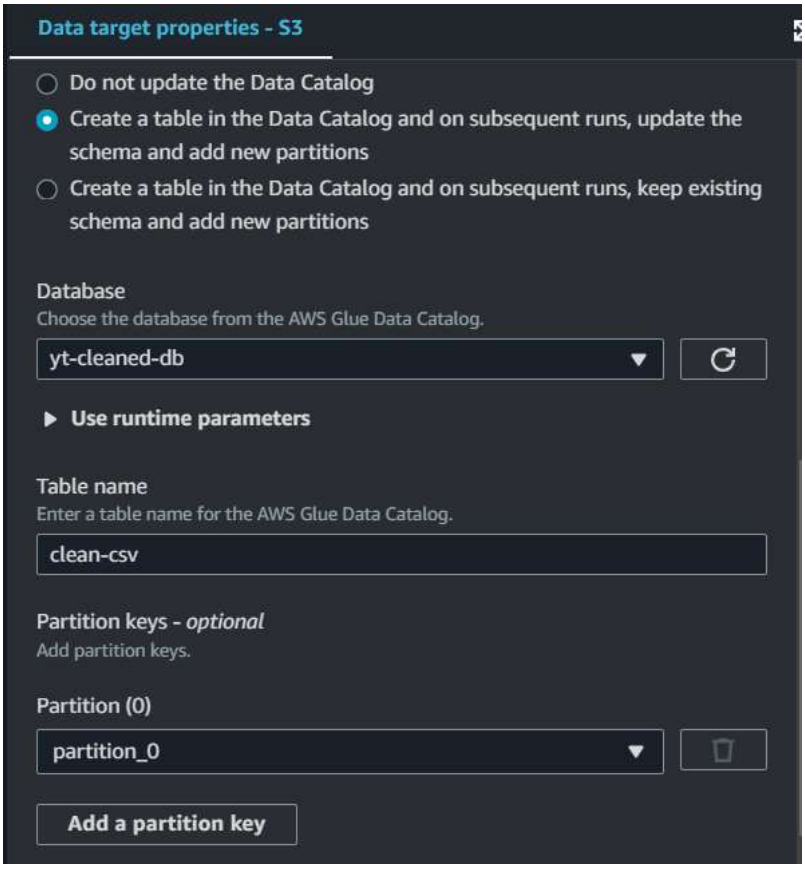


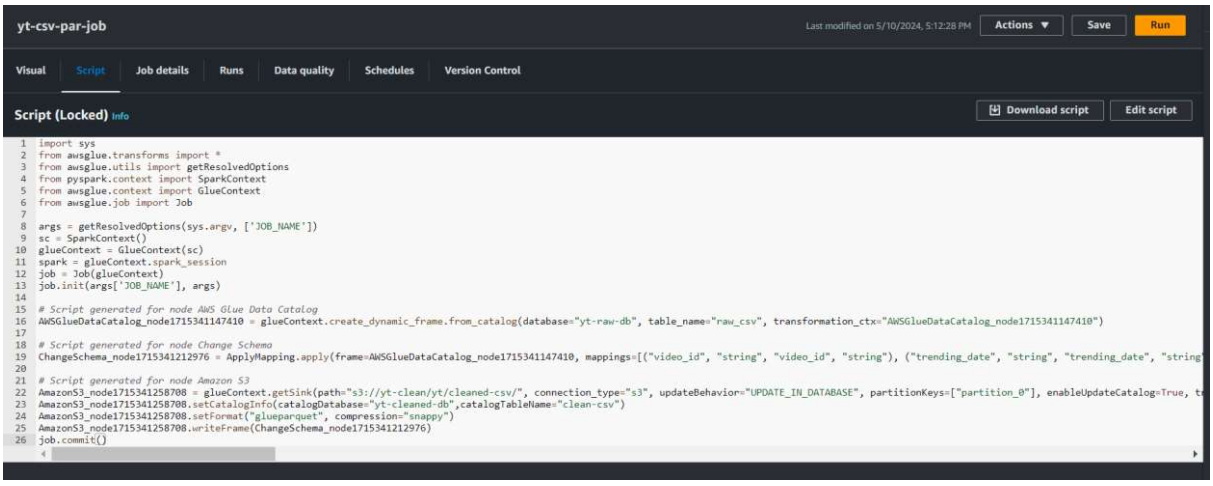
- Choose the Transformation that we need to apply, Here we are changing the schema of the table



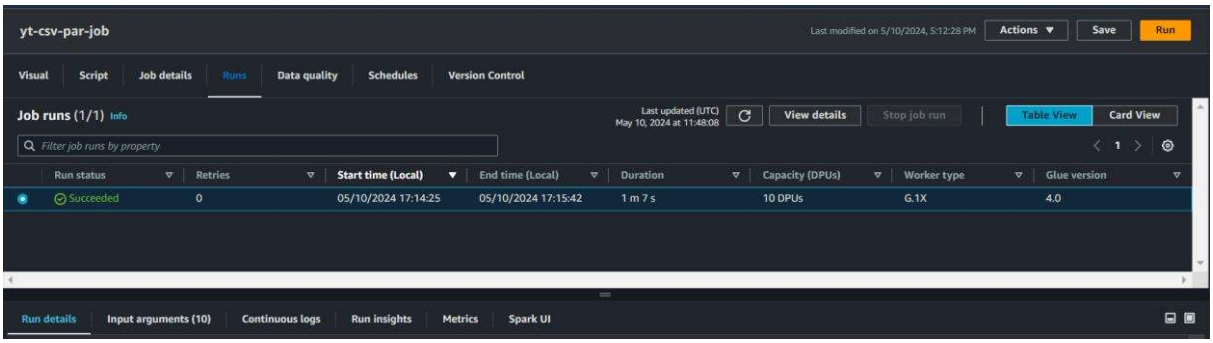
- Choose the Data Target, here we choose the cleansed S3 bucket for the parquets and cleansed database for the tables
- Also choose the format(parquet) for the conversion, Partition key (region : partition_0) and create the job


Visual
Script
Job details
Runs
Data quality
Schedules
Version Control
Data source - Data Catalog
AWS Glue Data Catalog
Transform - Change Sche...
Change Schema
Data target - S3 bucket
Amazon S3
Data target properties - S3
Name
Amazon S3
Node parents
Choose which nodes will provide inputs for this one.
Choose one or more parent node
Change Schema
ApplyMapping - Transform
Format
Parquet
Compression Type
Snappy
S3 Target Location
Choose an S3 location in the format s3://bucket/prefix/object/ with a trailing slash (/).
s3://yt-clean/yt/cleaned-
View
Browse S3
Data Catalog update options
Info

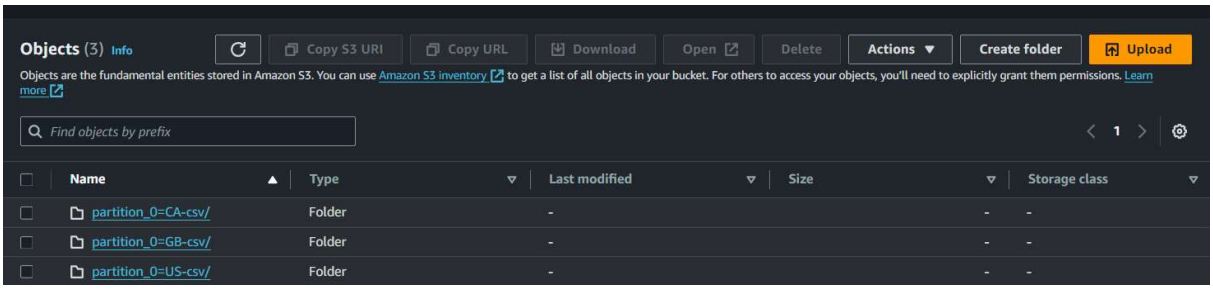

Data target properties - S3
Do not update the Data Catalog
Create a table in the Data Catalog and on subsequent runs, update the schema and add new partitions
Create a table in the Data Catalog and on subsequent runs, keep existing schema and add new partitions
Database
Choose the database from the AWS Glue Data Catalog.
yt-cleaned-db
Use runtime parameters
Table name
Enter a table name for the AWS Glue Data Catalog.
clean-csv
Partition keys - optional
Add partition keys.
Partition (0)
partition_0
Add a partition key


yt-csv-par-job
Last modified on 5/10/2024, 5:12:28 PM
Actions
Save
Run
Visual
Script
Job details
Runs
Data quality
Schedules
Version Control
Script (Locked) Info
Download script
Edit script
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
args = getResolvedOptions(sys.argv, ['JOB_NAME'])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)
# Script generated for node AWS Glue Data Catalog
AWSGlueDataCatalog_node1715341147410 = glueContext.create_dynamic_frame.from_catalog(database="yt-raw-db", table_name="raw_csv", transformation_ctx="AWSGlueDataCatalog_node1715341147410")
# Script generated for node Change Schema
ChangeSchema_node1715341212976 = ApplyMapping.apply(frame=AWSGlueDataCatalog_node1715341147410, mappings=[("video_id", "string", "video_id", "string"), ("trending_date", "string", "trending_date", "string
# Script generated for node Amazon S3
AmazonS3_node1715341258708 = glueContext.getSink(path="s3://yt-clean/yt/cleaned-csv/", connection_type="s3", updateBehavior="UPDATE_IN_DATABASE", partitionKeys=["partition_0"], enableUpdateCatalog=True, t
AmazonS3_node1715341258708.setCatalogInfo(catalogDatabase="yt-cleaned-db",catalogTableName="clean-csv")
AmazonS3_node1715341258708.setFormat("glueparquet", compression="snappy")
AmazonS3_node1715341258708.writeFrame(ChangeSchema_node1715341212976)
job.commit()

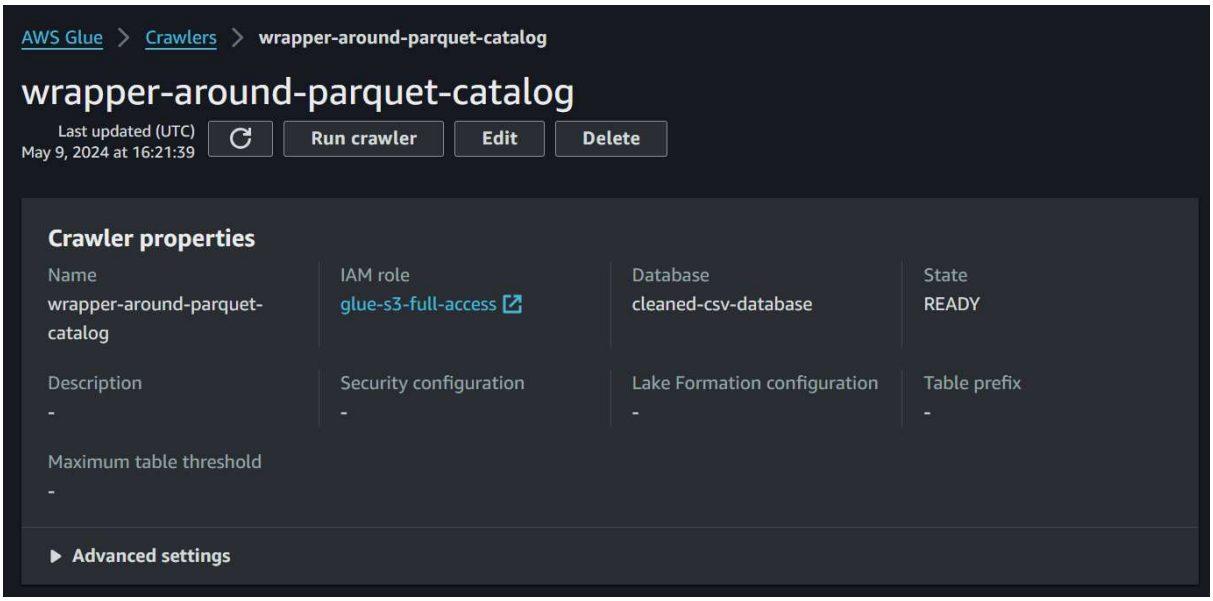
16. Run the ETL job.
    - It creates the parquets file in the s3 and tables in the cleaned database



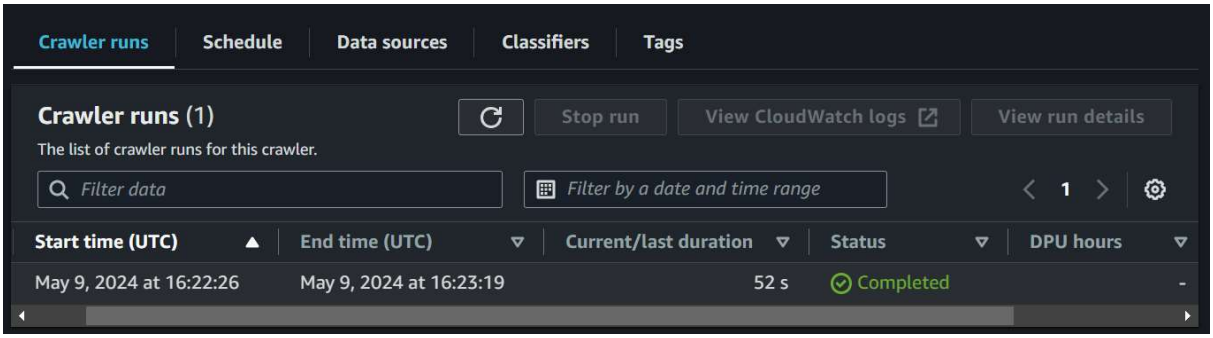
- We can verify that parquets are generated according to the partition key specified in the cleansed s3 bucket.



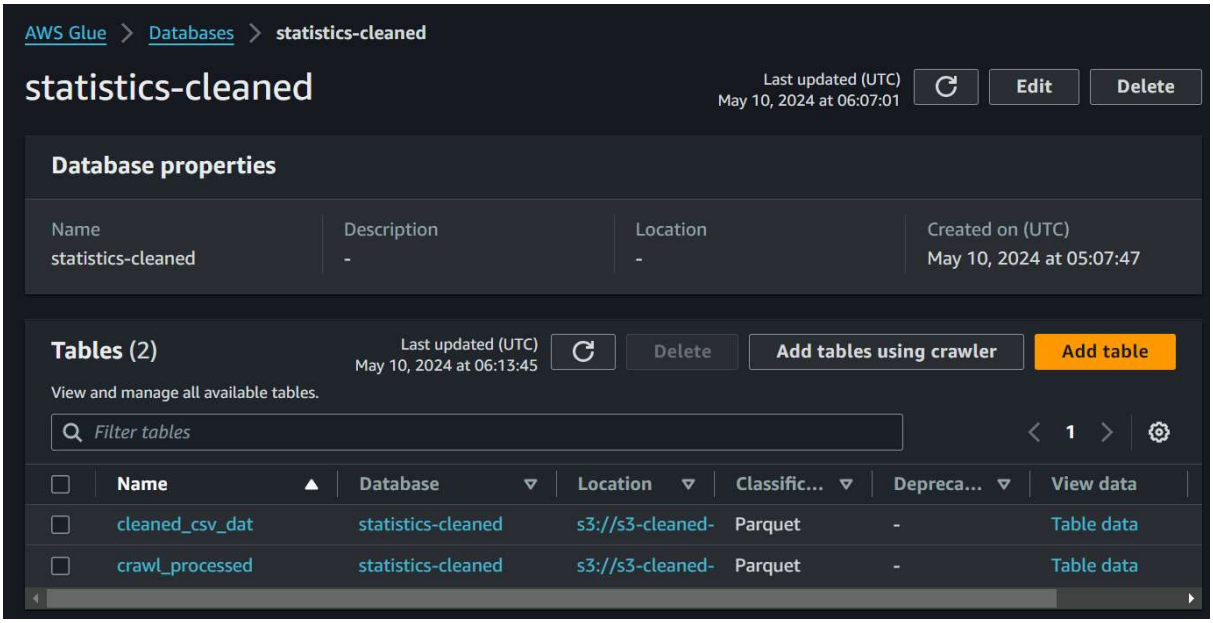
Now, I have created another crawler inorder to create a catalog like a wrapper around the newly created parquet files.



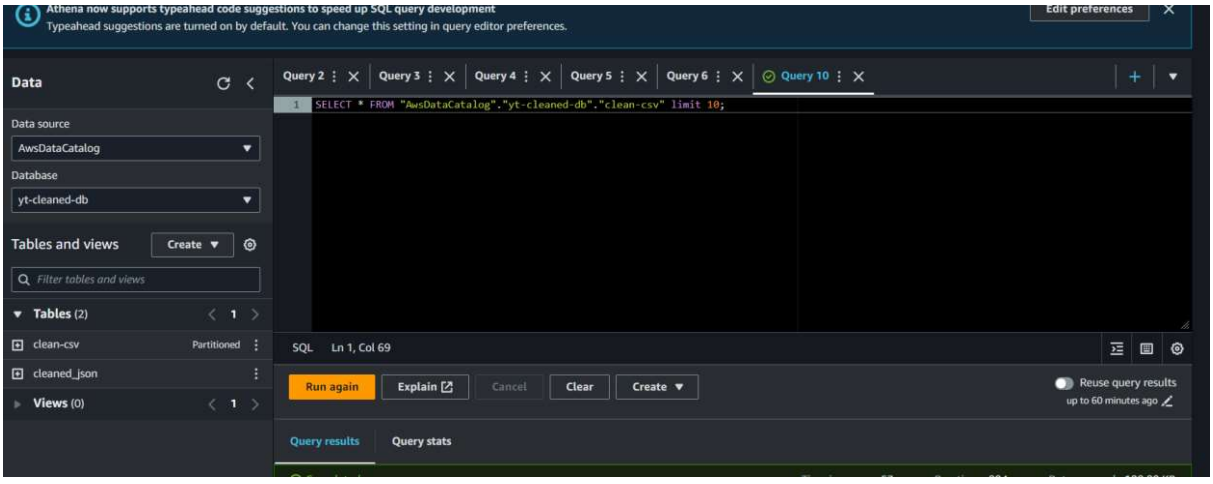
Then, click on run crawler. Then it will create a catalog table inside cleaned-csv-database

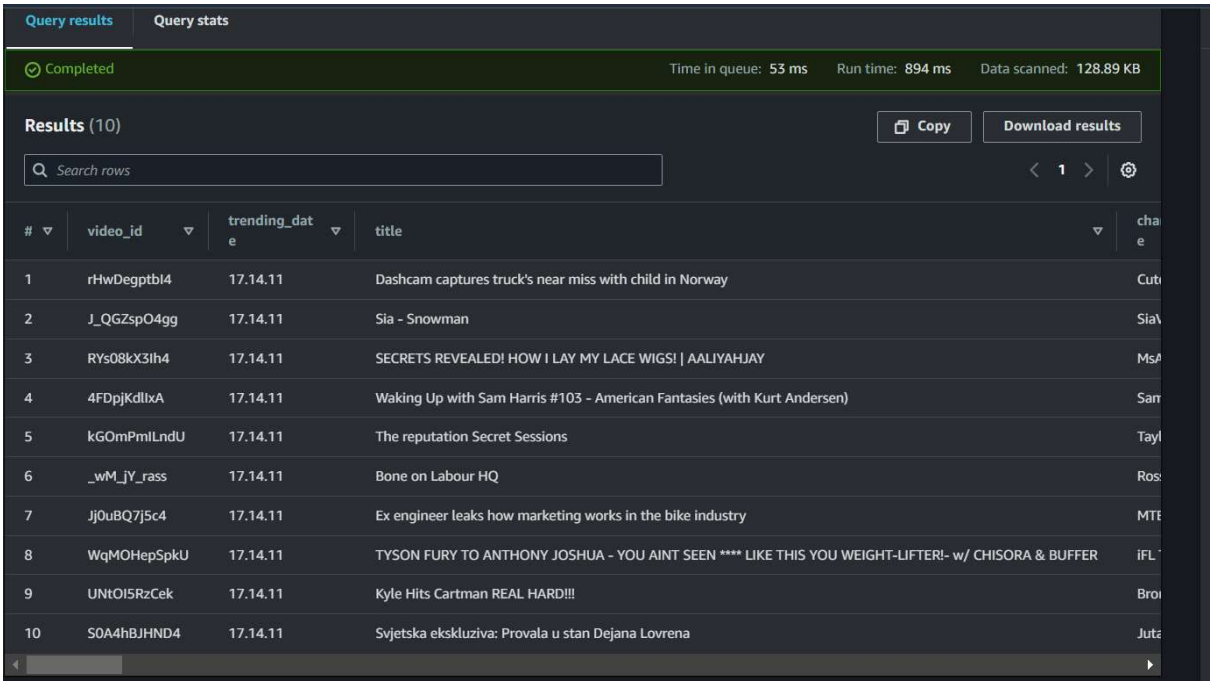


Now, the table will be created in the data catalog(cleaned_csv_dat), this table contains the data of the csv files for the 3 regions we specified earlier

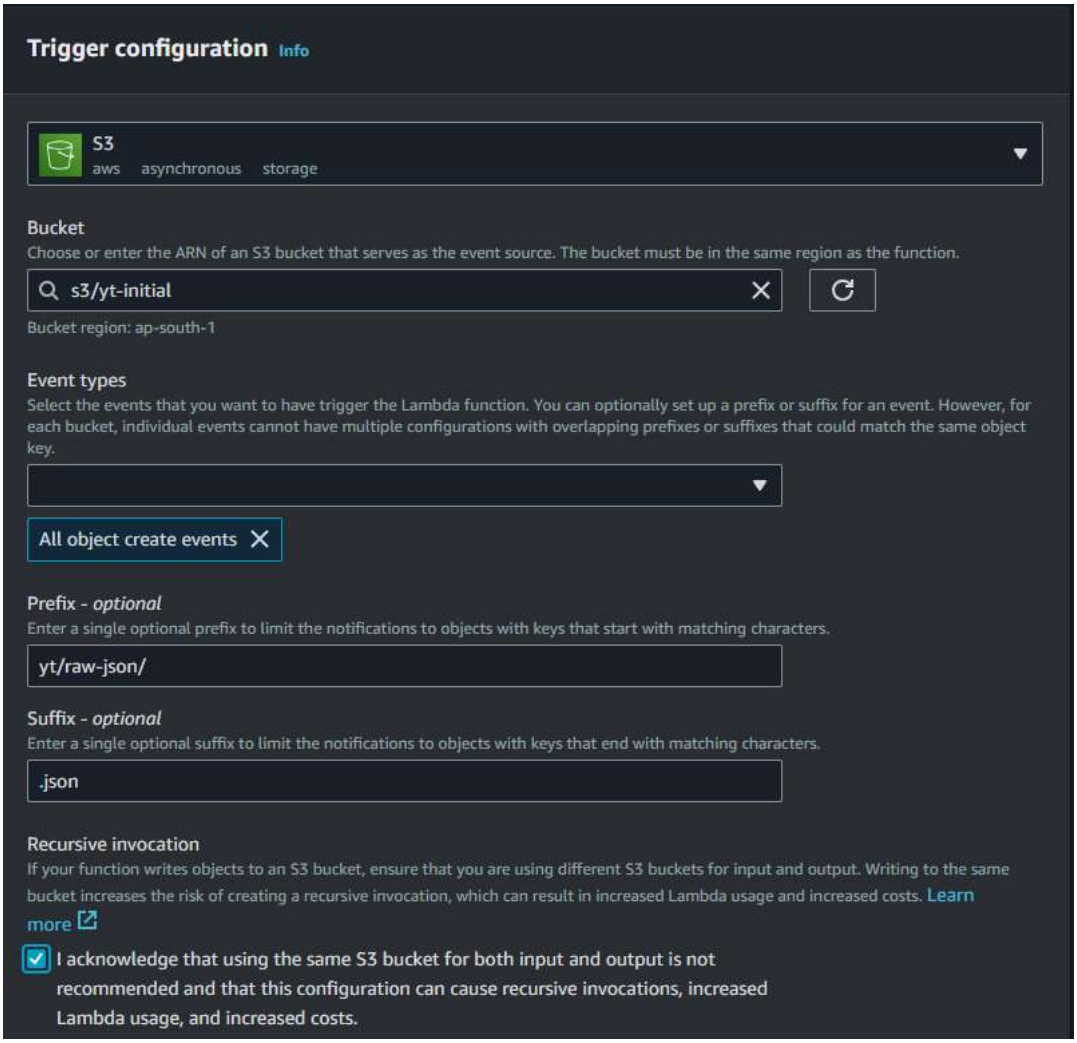


- Also we can verify that the partition table is generated and we now query it for further confirmation.

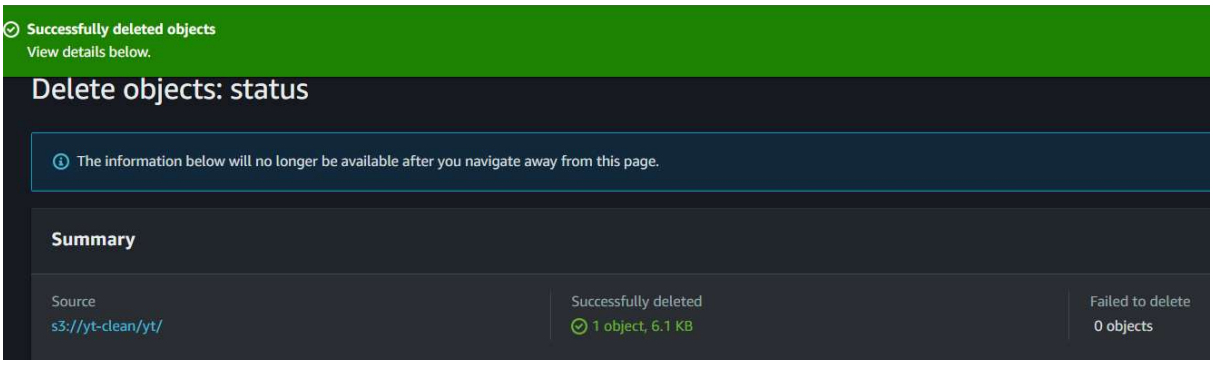




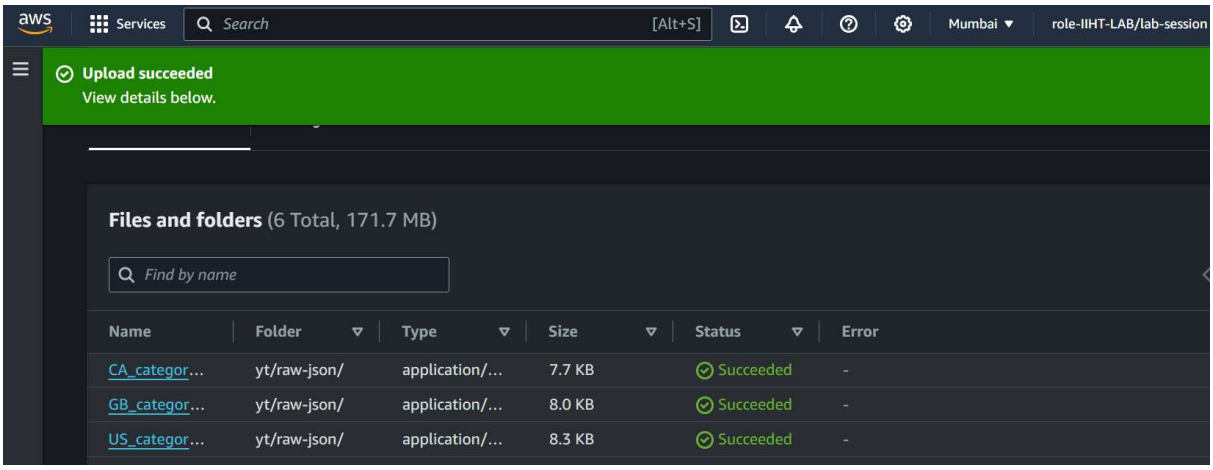
17. Create a Lambda Trigger so as to generate all parquets for json file
    - Initially we generated a single parquet file for single json file manually, to ensure the process that we follow is correct.
    - So, we need generate all parquet files for all json data. In order to do this we need to add a trigger to the lambda function,
    - The purpose of Adding trigger to lambda is to automate the generation of the parquets for all json files
    - so in order to trigger the lambda we delete all the uploaded json files and previously generated single parquet file, and reupload the files, now it triggers and generate the parquets.
    - Make sure provide all the details in the trigger details.



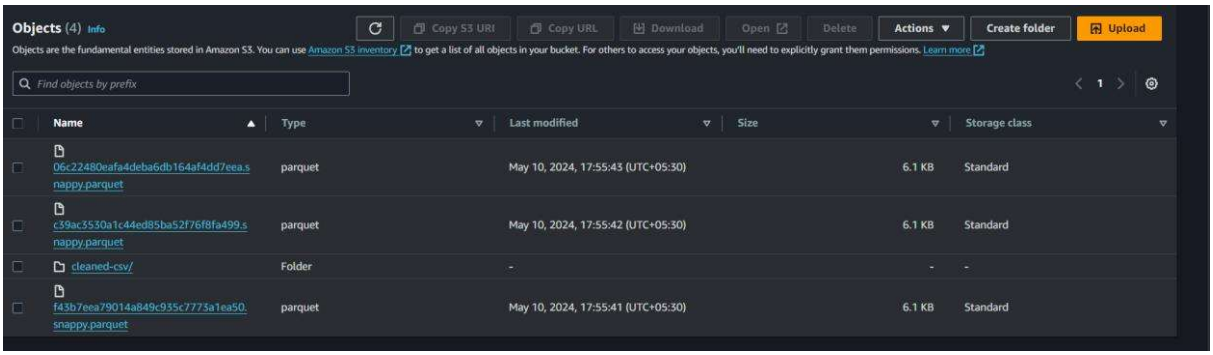
- deleting json files



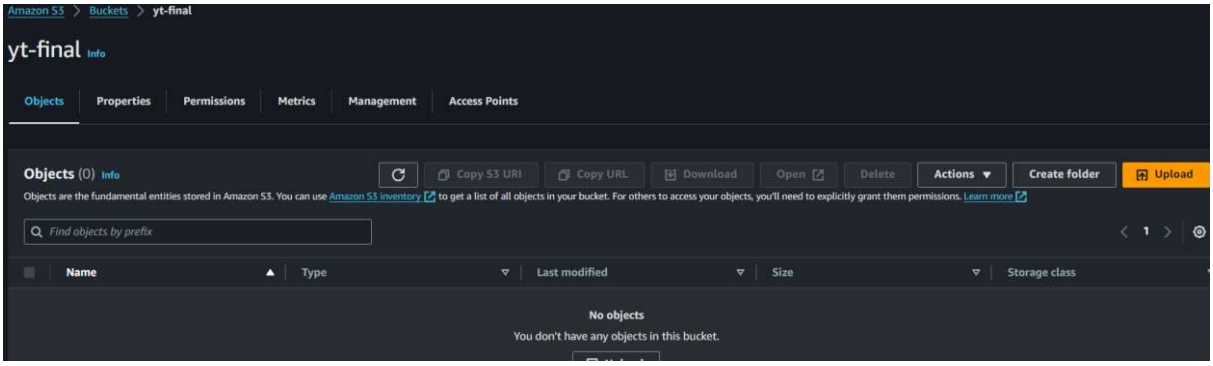
- Reupload the json files



- Confirmation for the parquet files generated

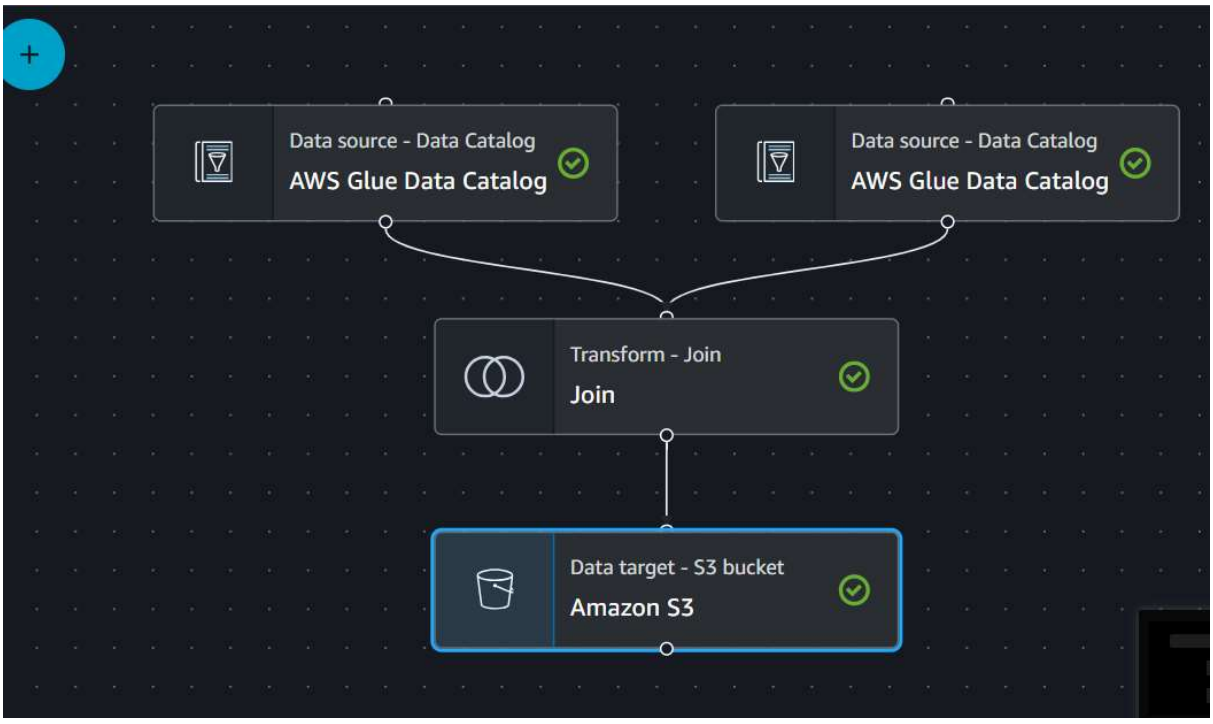


18. Create Final S3 Bucket for the Analytical purpose.

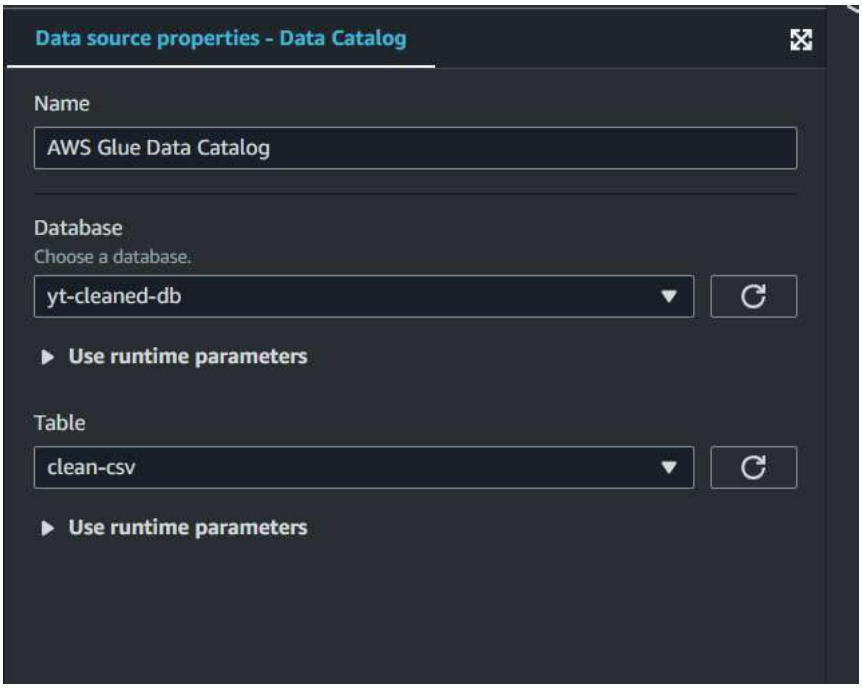


19. Create a ETL job combining the both json and csv Data catalogs
    - We create the Glue ETL job for combining the csv and json Data Catalog and store it in a new table, so that we can perform data visualization on the that single table using some visualisation tools.
    - We use Visual ETL for designing the job flow

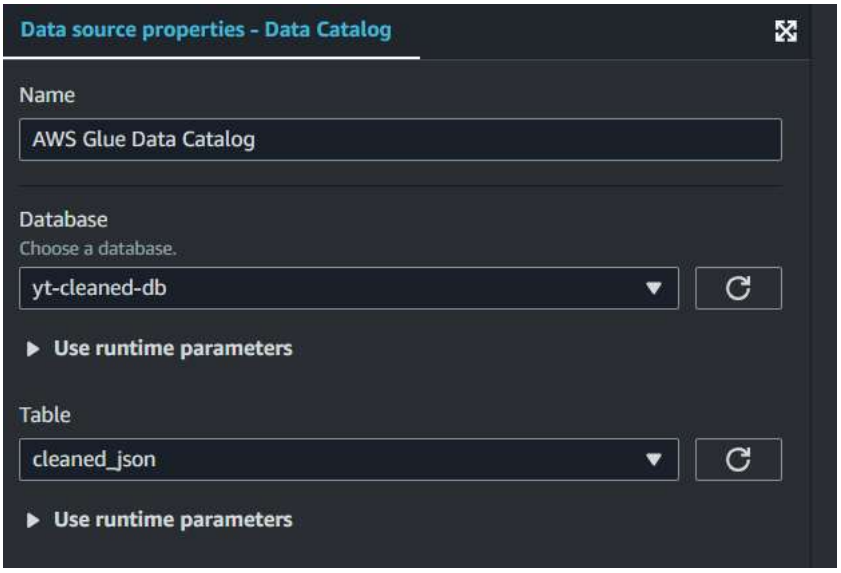
`



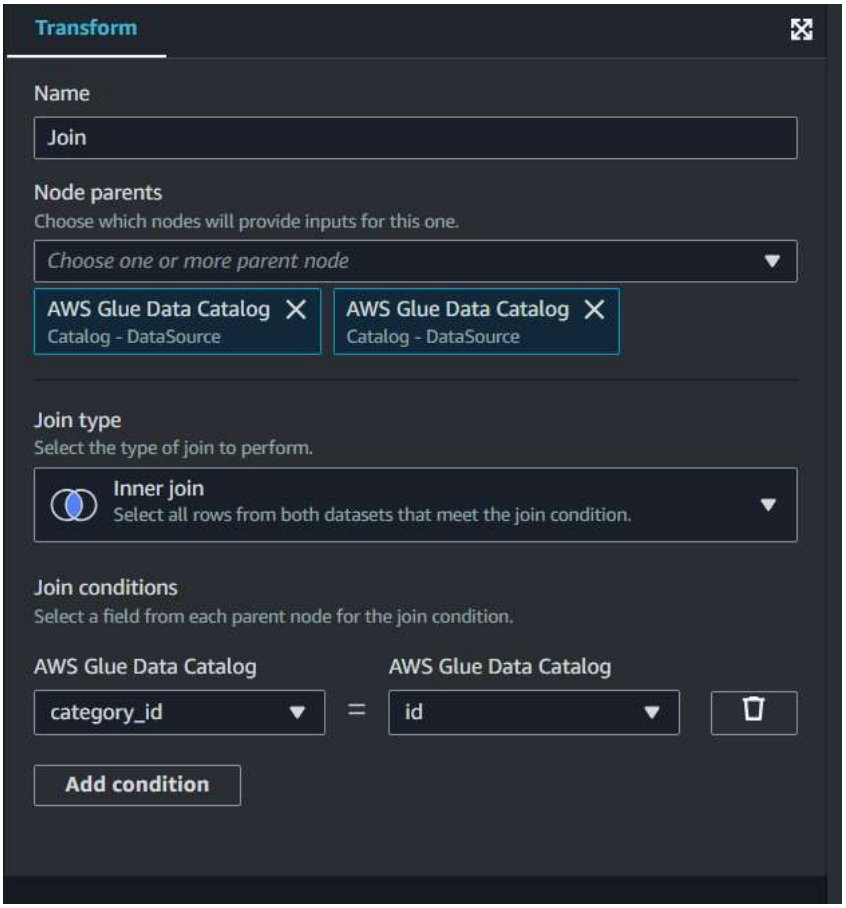
- Set Data source properties of csv data i.e. csv database and csv data table



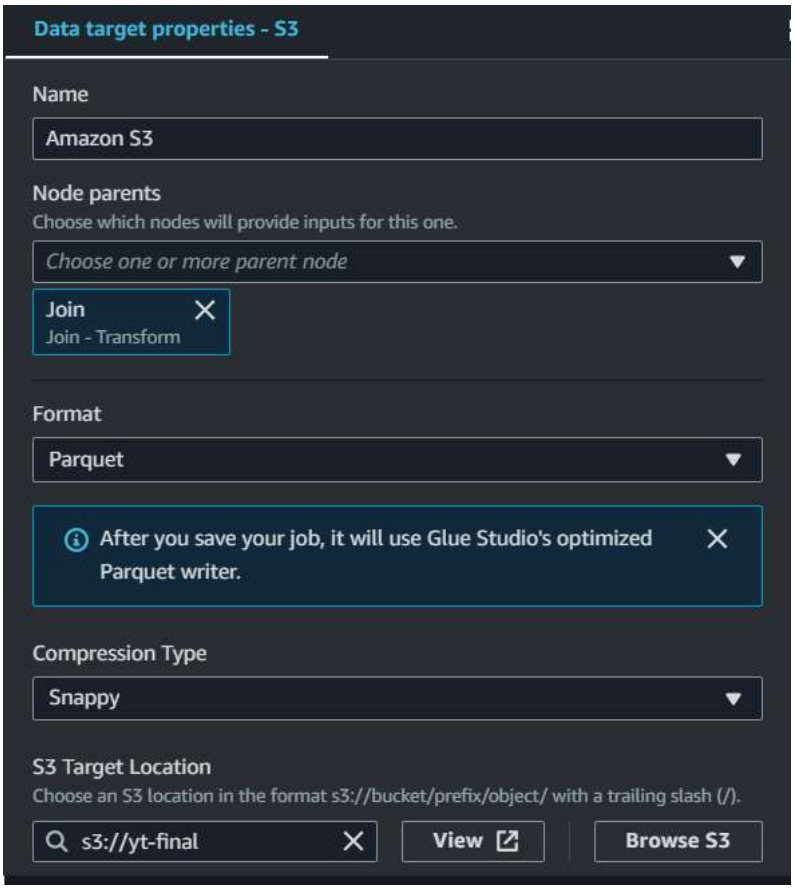
- Data source properties of json data i.e. database and json table data



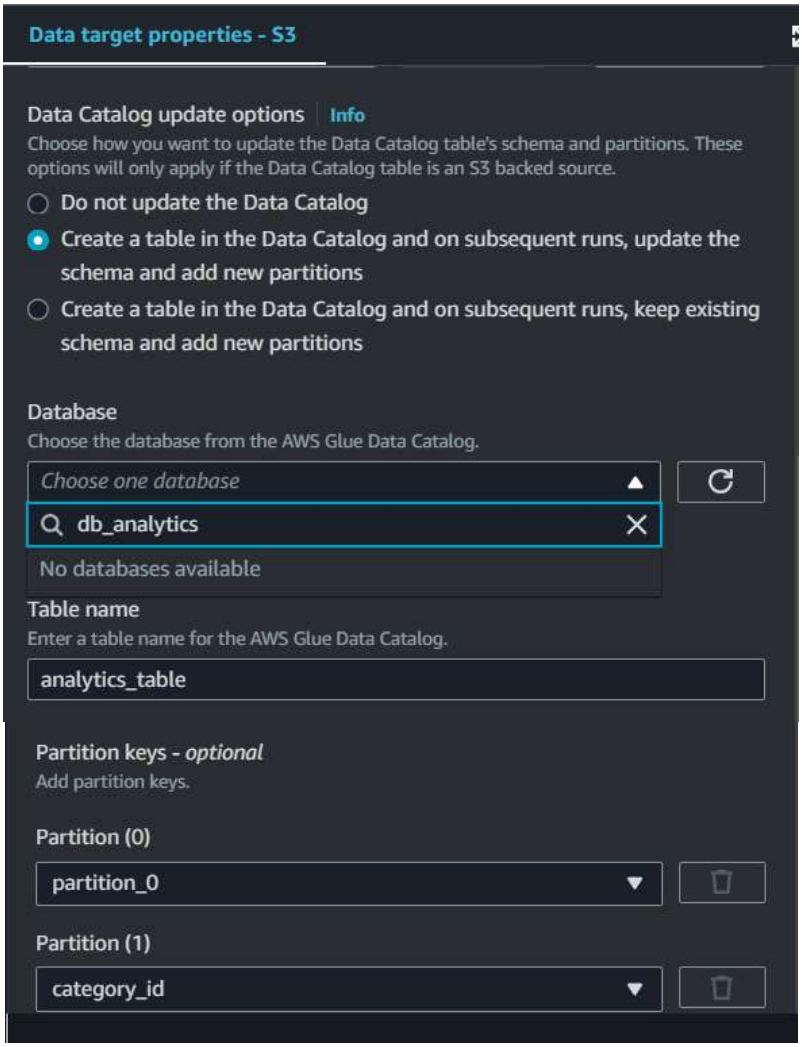
- Configure the Transformation, Here we inner join the both Json and csv Data catalogs
- Also provide the joining condition for the tables.



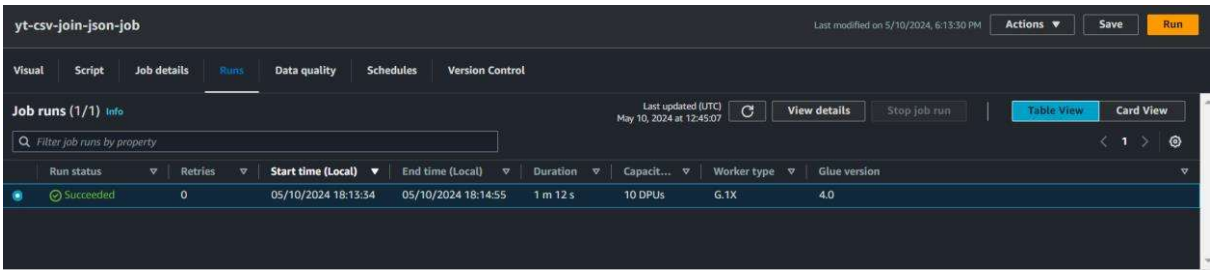
- Provide the Data target properties like, Format(Parquet) of the resultant and the location the final s3 bucket for storing the final result.



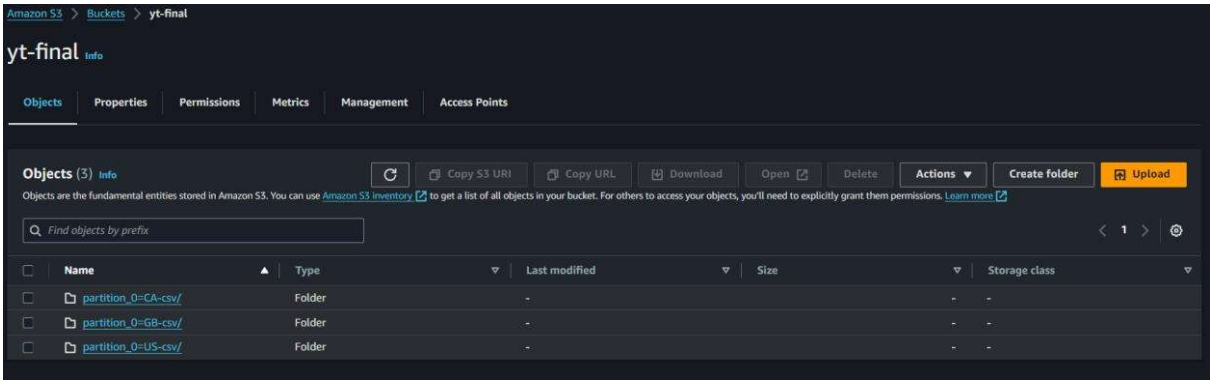
- Also choose the option for creating new table and write the updated schema
- Also provide Database and table details where new data to be stored
- Provide the partition key, here region as primary partition key and category_id as the secondary partition key.



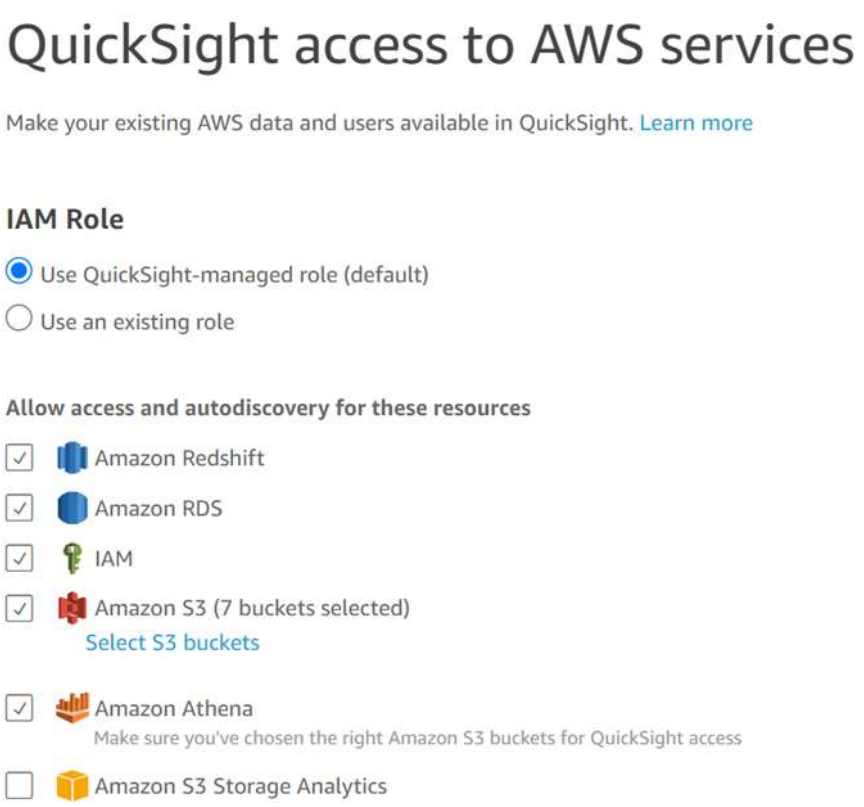
20. Run the ETL job
    - Successful run of the ETL job confirms you that new table is generated in the final database



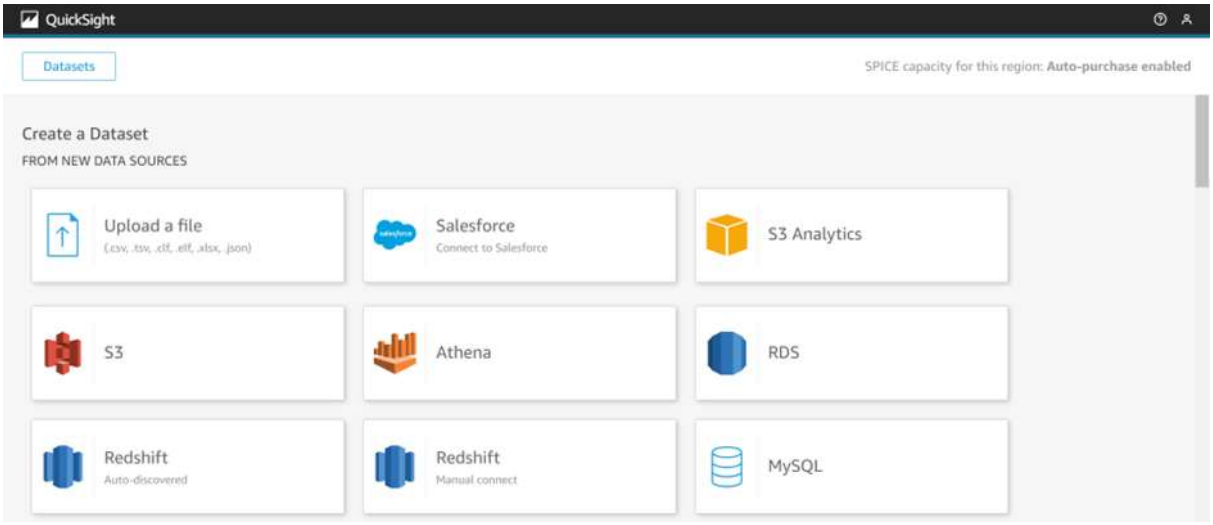
    - Verify the final parquet are generated in the final S3 bucket in accordance with the partition keys specified.



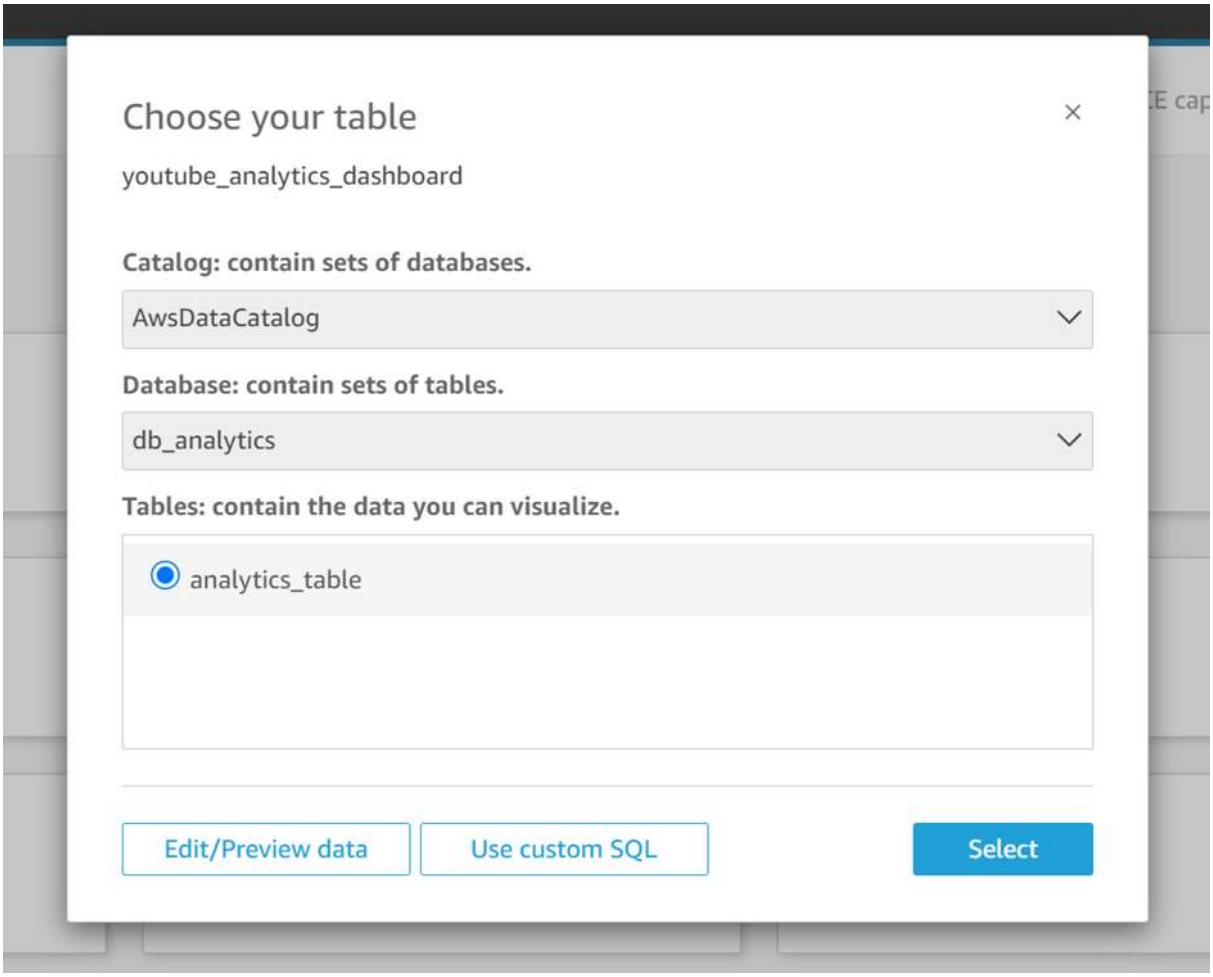
21. Visual Analysis using the AWS Quick Sight
    - Give, Quick Sight access to AWS Services.



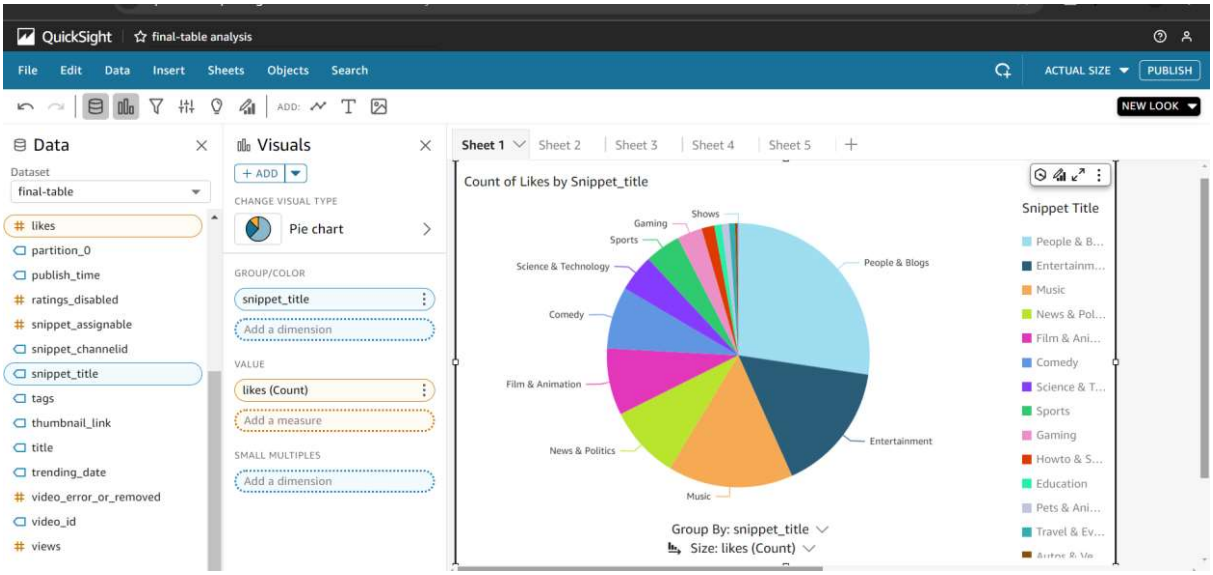
    - Choose dataset from the Aws Athena as data source.

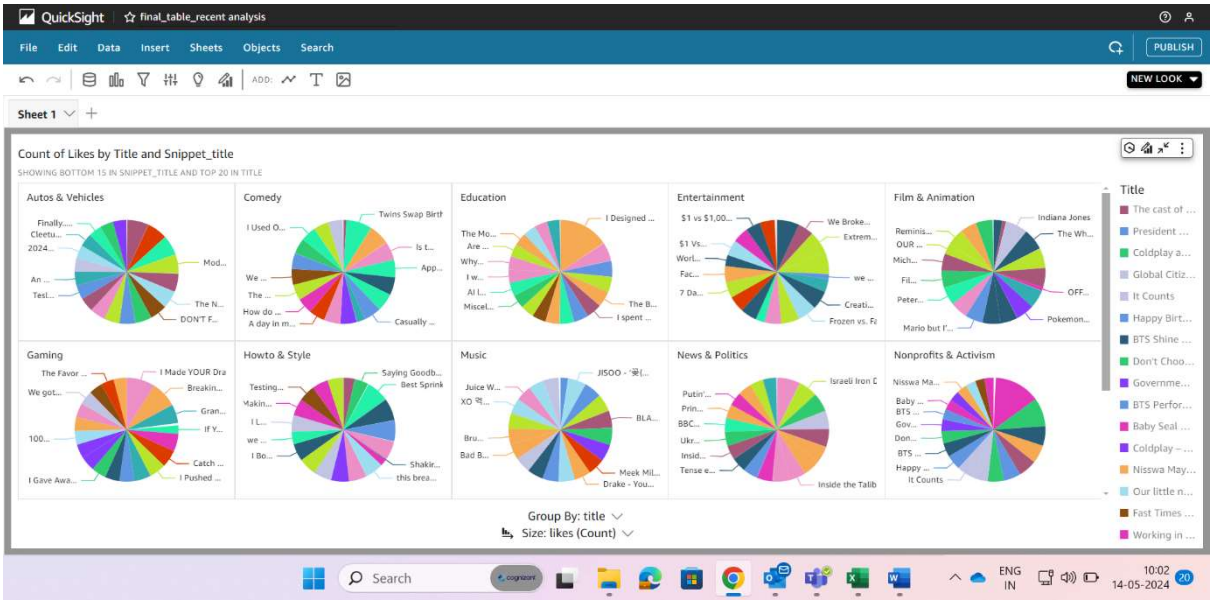


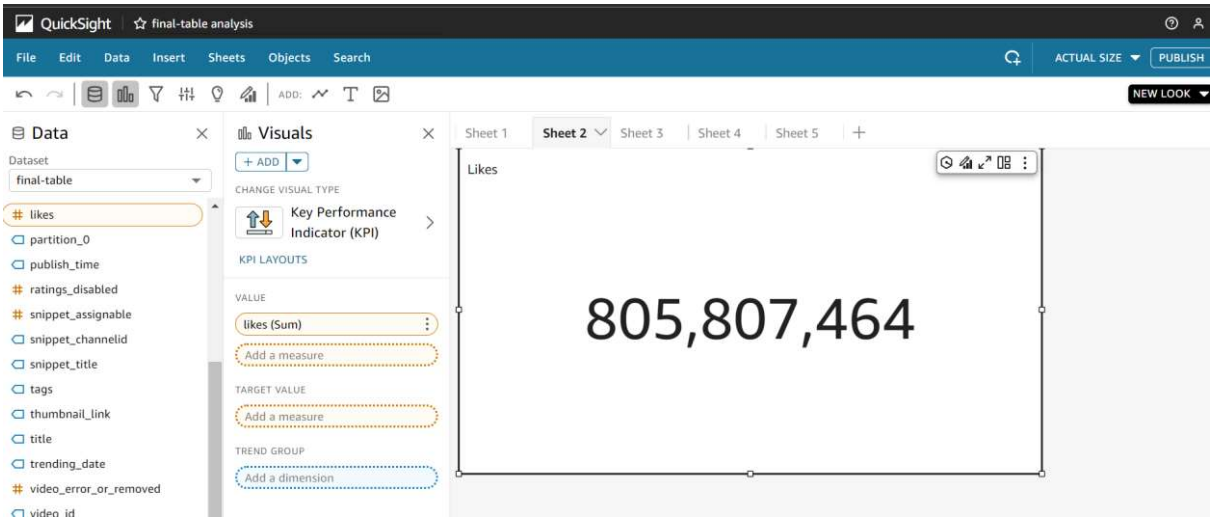
- Choose the database and the Table for the Visualisation of the data and choose Edit/Preview and proceed for the analysis.

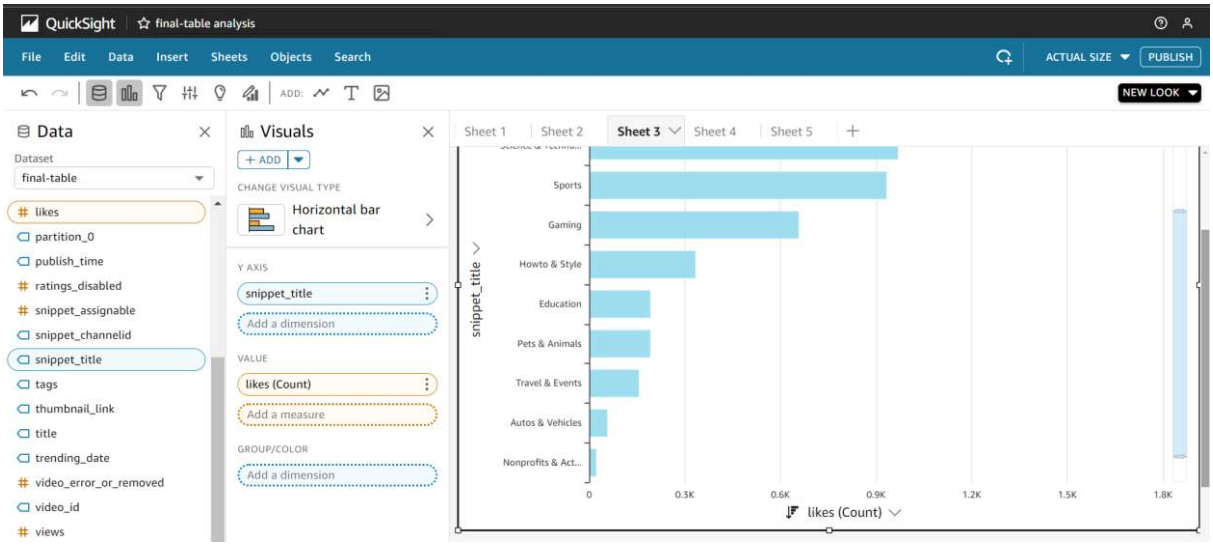


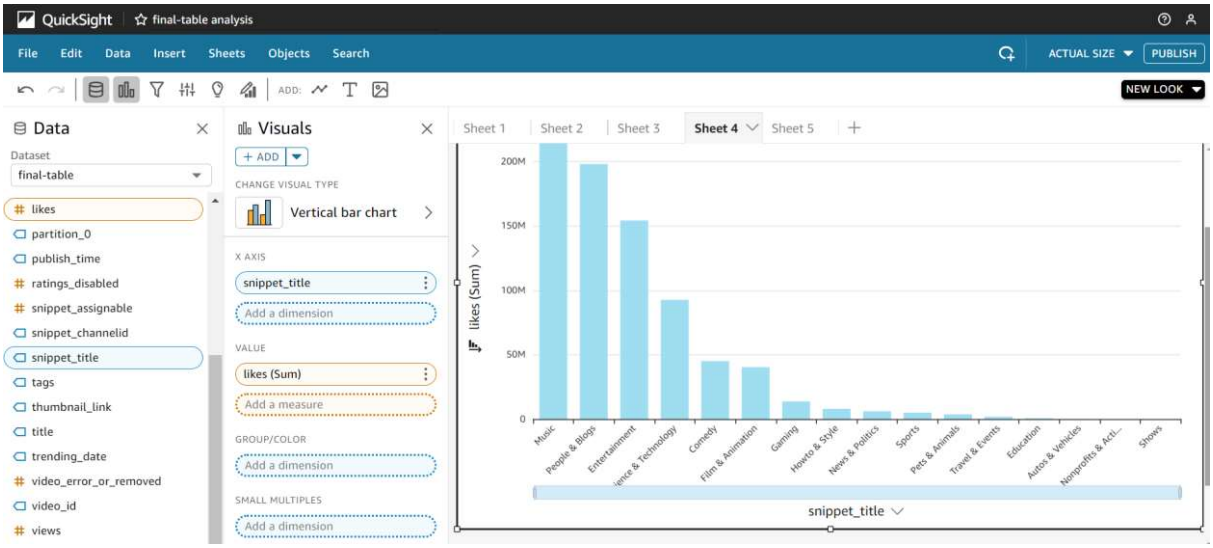
- Choose Different Visual styles and choose appropriate attributes for output to generate.


QuickSight
final-table analysis
File Edit Data Insert Sheets Objects Search
ACTUAL SIZE
PUBLISH
NEW LOOK
Data
Dataset
final-table
likes
partition_0
publish_time
ratings_disabled
snippet_assignable
snippet_channelid
snippet_title
tags
thumbnail_link
title
trending_date
video_error_or_removed
video_id
views
Visuals
+ ADD
CHANGE VISUAL TYPE
Pie chart
GROUP/COLOR
snippet_title
Add a dimension
VALUE
likes (Count)
Add a measure
SMALL MULTIPLES
Add a dimension
Sheet 1
Sheet 2
Sheet 3
Sheet 4
Sheet 5
Count of Likes by Snippet_title
Shows
Gaming
Sports
Science & Technology
Comedy
Film & Animation
News & Politics
Music
People & Blogs
Entertainment
Snippet Title
People & B...
Entertainm...
Music
News & Pol...
Film & Ani...
Comedy
Science & T...
Sports
Gaming
Howto & S...
Education
Pets & Ani...
Travel & Ev...
Group By: snippet_title
Size: likes (Count)


QuickSight
final_table_recent analysis
File Edit Data Insert Sheets Objects Search
PUBLISH
NEW LOOK
Sheet 1
Count of Likes by Title and Snippet_title
SHOWING BOTTOM 15 IN SNIPPET_TITLE AND TOP 20 IN TITLE
Autos & Vehicles
Comedy
Education
Entertainment
Film & Animation
Gaming
Howto & Style
Music
News & Politics
Nonprofits & Activism
Title
The cast of ...
President ...
Coldplay a...
Global Citiz...
It Counts
Happy Birt...
BTS Shine ...
Don't Choo...
Governme...
BTS Perfor...
Baby Seal ...
Coldplay – ...
Nisswa May...
Our little n...
Fast Times ...
Working in ...
Group By: title
Size: likes (Count)
Search
ENG
IN
10:02
14-05-2024


QuickSight
final-table analysis
File Edit Data Insert Sheets Objects Search
ACTUAL SIZE
PUBLISH
NEW LOOK
Data
Dataset
final-table
likes
partition_0
publish_time
ratings_disabled
snippet_assignable
snippet_channelid
snippet_title
tags
thumbnail_link
title
trending_date
video_error_or_removed
video_id
Visuals
+ ADD
CHANGE VISUAL TYPE
Key Performance Indicator (KPI)
KPI LAYOUTS
VALUE
likes (Sum)
Add a measure
TARGET VALUE
Add a measure
TREND GROUP
Add a dimension
Sheet 1
Sheet 2
Sheet 3
Sheet 4
Sheet 5
Likes
805,807,464





Conclusion : Data Analysis and Visualisation is successfully done on the YouTube Trending data using different AWS Services.