

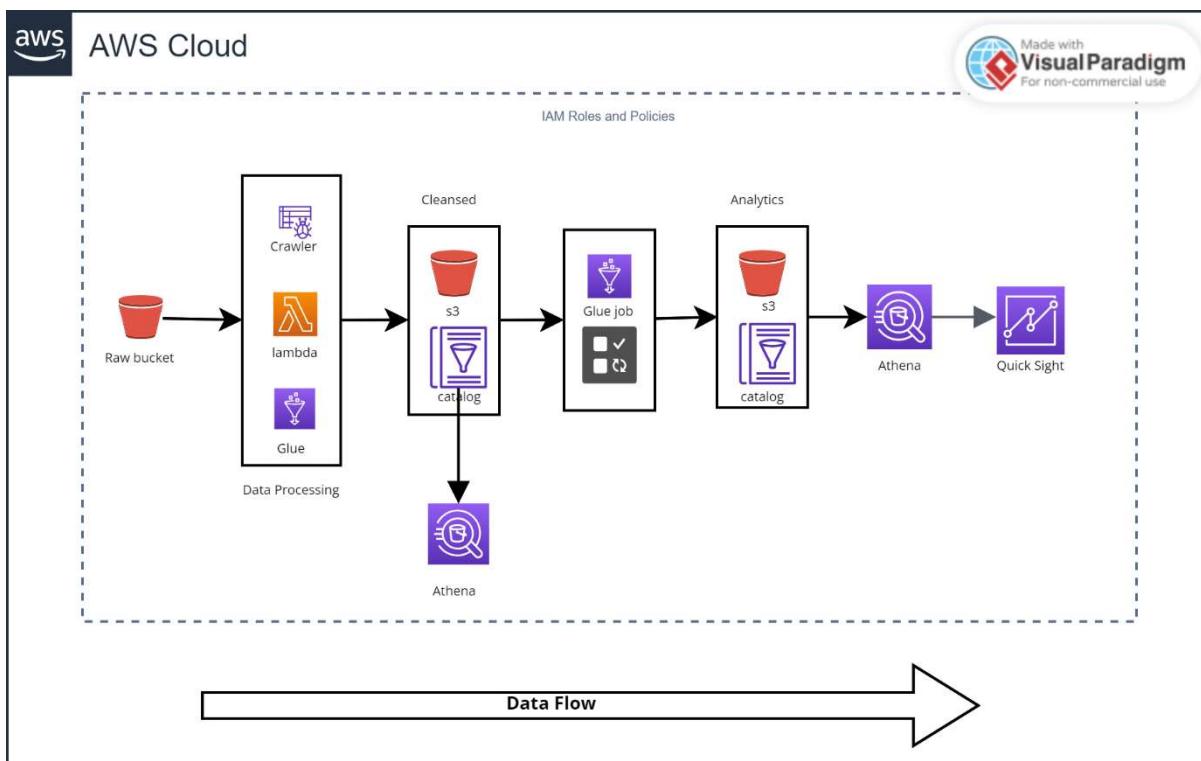
Data Analysis and Visualisation on Trending YouTube Data using AWS Glue and QuickSight

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.

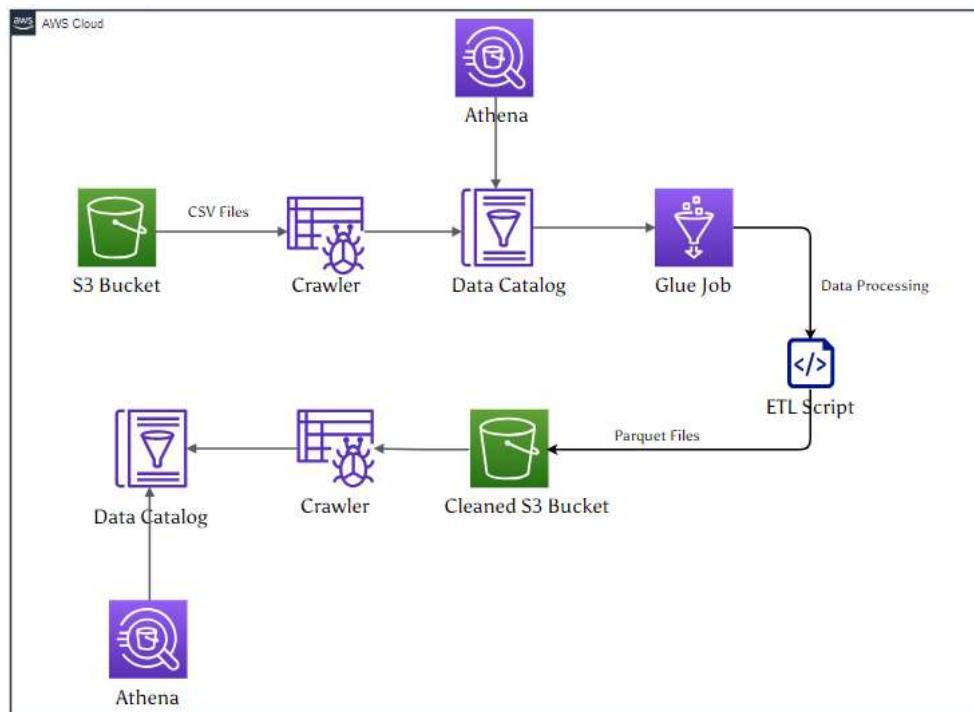
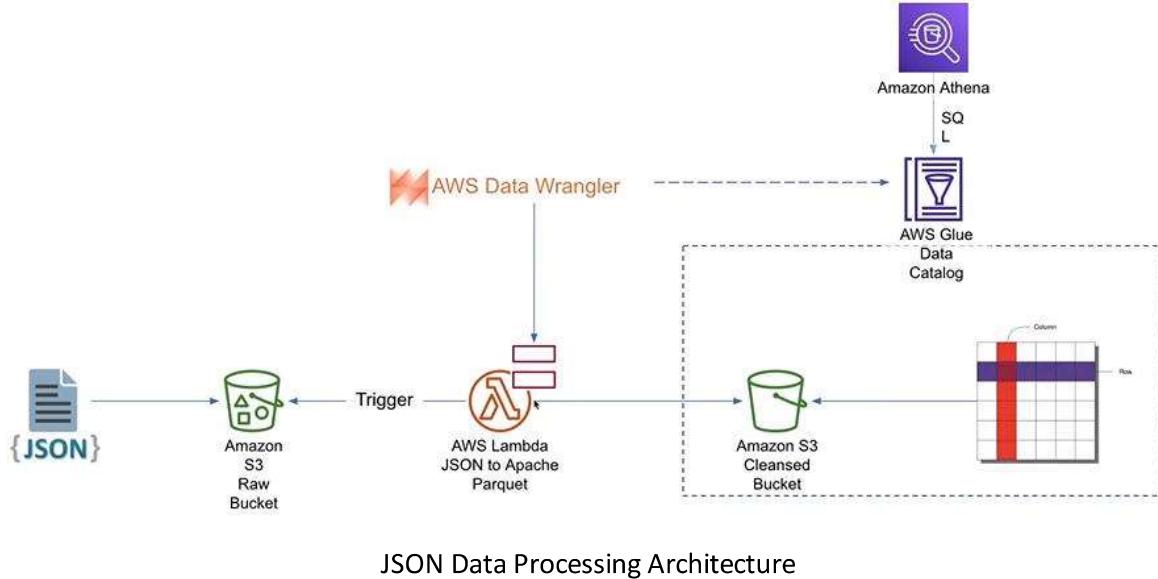
Dataset : The data is about 211MB, in which it consists of two types of files one is json files and other is csv files. There are total 6 files in which 3 are json files and 3 are csv files.

Architecture



Data Cleansing

Semi-structured data to Structured pipeline



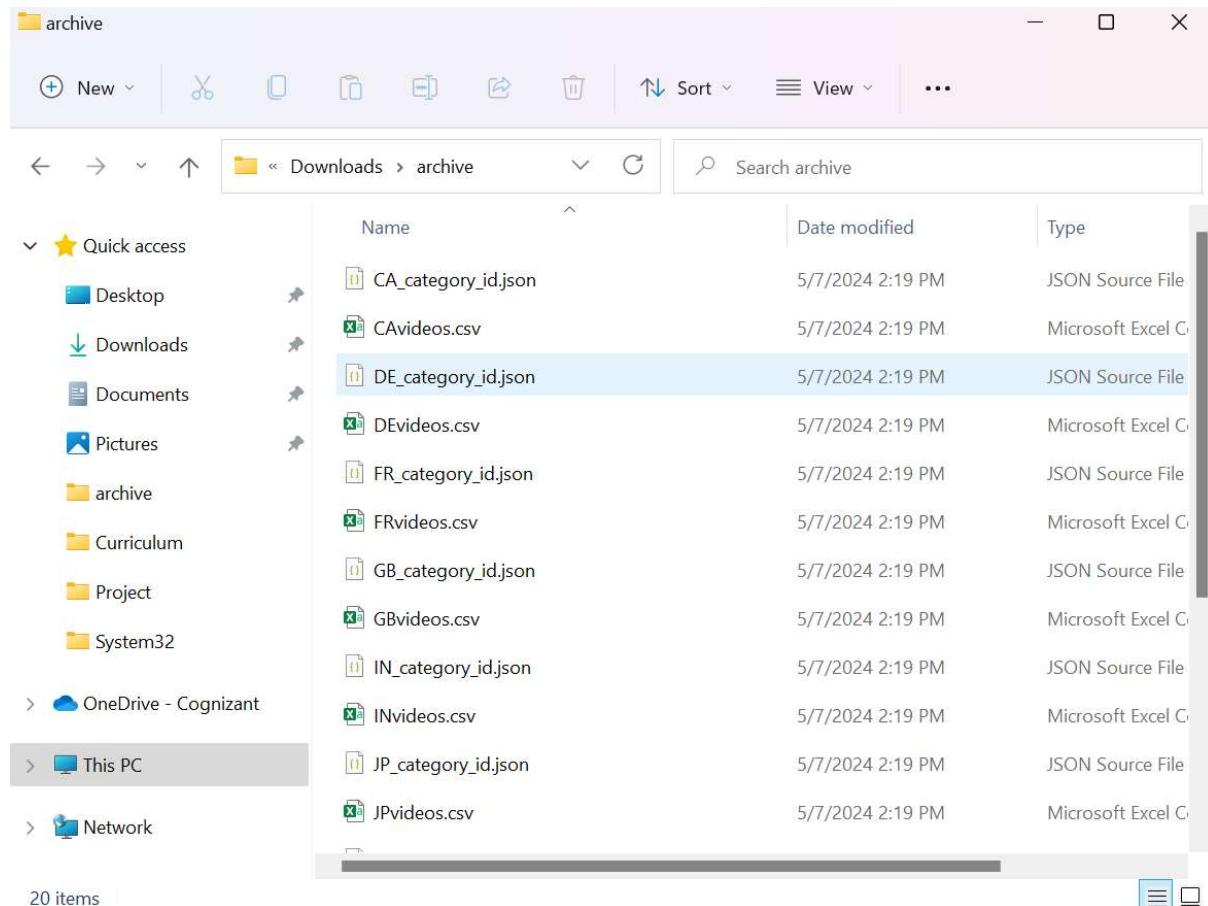
CSV Data Processing Architecture

IMPLEMENTATION

1. Download the trending Youtube data from Kaggle

The screenshot shows the Kaggle interface. On the left, there's a sidebar with options like 'Create', 'Home', 'Competitions', 'Datasets' (which is selected), 'Models', 'Code', 'Discussions', 'Learn', and 'More'. The main area displays a dataset titled 'Trending YouTube Video Statistics' by 'MITCHELL J - UPDATED 5 YEARS AGO'. It has 5398 rows and a 'Download (211 MB)' button. Below the title, it says 'Daily statistics for trending YouTube videos'. There are tabs for 'Data Card', 'Code (3080)', 'Discussion (44)', and 'Suggestions (1)'. To the right, there's a large 'YouTube' logo. At the bottom, there's an 'About Dataset' section, 'Usability' rating (7.94), and a 'License' link.

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



3. Login to your AWS account

The screenshot shows the AWS Console Home page. On the left, there's a sidebar with 'Recently visited' services: Athena, AWS Organizations, AWS Glue, Service Quotas, S3, CloudShell, Lambda, CodePipeline, EC2, IAM, and Resource Access Manager. On the right, there's a 'Applications' section with a count of 0. It includes a 'Create application' button and a note: 'No applications. Get started by creating an application.' Below this is another 'Create application' button.

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

The screenshot shows the AWS S3 service 'Account snapshot' page. It displays two 'General purpose buckets': 'cleaned-bucket-etl-pipeline' and 'youtube-raw-data-etl-analysis'. Both buckets were created on May 9, 2024, at different times (08:10:05 and 07:57:38 UTC+05:30). There are buttons for 'Copy ARN', 'Empty', 'Delete', and 'Create bucket'.

Name	AWS Region	IAM Access Analyzer	Creation date
cleaned-bucket-etl-pipeline	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	May 9, 2024, 08:10:05 (UTC+05:30)
youtube-raw-data-etl-analysis	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	May 9, 2024, 07:57:38 (UTC+05:30)

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

The screenshot shows the AWS S3 console interface for the 'youtube-raw-data-etl-analysis' bucket. The 'Objects' tab is selected, showing two items: 'csv_dat/' and 'json_dat/'. Both are listed as 'Folder'. The table includes columns for Name, Type, Last modified, Size, and Storage class.

Name	Type	Last modified	Size	Storage class
csv_dat/	Folder	-	-	-
json_dat/	Folder	-	-	-

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

The screenshot shows the AWS S3 console interface for the 'json_files/' folder within the 'youtube-raw-data-etl-analysis' bucket. The 'Objects' tab is selected, showing three objects: 'CA_category_id.json', 'GB_category_id.json', and 'US_category_id.json'. All three are listed as 'json' files. The table includes columns for Name, Type, Last modified, Size, and Storage class.

Name	Type	Last modified	Size	Storage class
CA_category_id.json	json	May 9, 2024, 15:46:25 (UTC+05:30)	7.7 KB	Standard
GB_category_id.json	json	May 9, 2024, 15:46:25 (UTC+05:30)	8.0 KB	Standard
US_category_id.json	json	May 9, 2024, 15:46:25 (UTC+05:30)	8.3 KB	Standard

- 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>

The screenshot shows the AWS S3 console interface for the 'csv_files/' folder within the 'youtube-raw-data-etl-analysis' bucket. The 'Objects' tab is selected, showing three objects: 'region=ca/', 'region=gb/', and 'region=us/'. All three are listed as 'Folder'. The table includes columns for Name, Type, Last modified, Size, and Storage class.

Name	Type	Last modified	Size	Storage class
region=ca/	Folder	-	-	-
region=gb/	Folder	-	-	-
region=us/	Folder	-	-	-

Objects (1) [Info](#)

Name	Type	Last modified	Size	Storage class
CAvideos.csv	csv	May 9, 2024, 15:52:01 (UTC+05:30)	61.1 MB	Standard

8. Create a Crawler to process JSON data

- Crawler is a glue service used to crawl data from different sources (S3) and stores the data in the glue catalog which further used for the analytical purpose.
- Give a name to the Crawler : yt-json-crawler
- Provide the Data Source location 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

Summary

Creation date	ARN
May 09, 2024, 22:44 (UTC+05:30)	arn:aws:iam::637423175804:role/yt-glue-role
Last activity	Maximum session duration
12 hours ago	1 hour

Permissions | Trust relationships | Tags | Access Advisor | Revoke sessions

Permissions policies (2) [Info](#)

Policy name	Type	Attached entities
AmazonS3FullAccess	AWS managed	2
AWSGlueServiceRole	AWS managed	2

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

AWS Glue > Databases > yt-raw-db

yt-raw-db

Last updated (UTC) May 10, 2024 at 07:17:40 [Edit](#) [Delete](#)

Database properties

Name yt-raw-db	Description -	Location -	Created on (UTC) May 10, 2024 at 07:17:31
-------------------	------------------	---------------	--

Tables (0)

Last updated (UTC) May 10, 2024 at 07:17:42 [Edit](#) [Delete](#) [Add tables using crawler](#) [Add table](#)

View and manage all available tables.

<input type="checkbox"/>	Name	Database	Location	Classification	Deprecation	View data	Data quality
--------------------------	------	----------	----------	----------------	-------------	---------------------------	--------------

[Filter tables](#) | [1](#) | [2](#) | [3](#) | [4](#) | [5](#) | [6](#) | [7](#) | [8](#) | [9](#) | [10](#) | [11](#) | [12](#) | [13](#) | [14](#) | [15](#) | [16](#) | [17](#) | [18](#) | [19](#) | [20](#) | [21](#) | [22](#) | [23](#) | [24](#) | [25](#) | [26](#) | [27](#) | [28](#) | [29](#) | [30](#) | [31](#) | [32](#) | [33](#) | [34](#) | [35](#) | [36](#) | [37](#) | [38](#) | [39](#) | [40](#) | [41](#) | [42](#) | [43](#) | [44](#) | [45](#) | [46](#) | [47](#) | [48](#) | [49](#) | [50](#) | [51](#) | [52](#) | [53](#) | [54](#) | [55](#) | [56](#) | [57](#) | [58](#) | [59](#) | [60](#) | [61](#) | [62](#) | [63](#) | [64](#) | [65](#) | [66](#) | [67](#) | [68](#) | [69](#) | [70](#) | [71](#) | [72](#) | [73](#) | [74](#) | [75](#) | [76](#) | [77](#) | [78](#) | [79](#) | [80](#) | [81](#) | [82](#) | [83](#) | [84](#) | [85](#) | [86](#) | [87](#) | [88](#) | [89](#) | [90](#) | [91](#) | [92](#) | [93](#) | [94](#) | [95](#) | [96](#) | [97](#) | [98](#) | [99](#) | [100](#) | [101](#) | [102](#) | [103](#) | [104](#) | [105](#) | [106](#) | [107](#) | [108](#) | [109](#) | [110](#) | [111](#) | [112](#) | [113](#) | [114](#) | [115](#) | [116](#) | [117](#) | [118](#) | [119](#) | [120](#) | [121](#) | [122](#) | [123](#) | [124](#) | [125](#) | [126](#) | [127](#) | [128](#) | [129](#) | [130](#) | [131](#) | [132](#) | [133](#) | [134](#) | [135](#) | [136](#) | [137](#) | [138](#) | [139](#) | [140](#) | [141](#) | [142](#) | [143](#) | [144](#) | [145](#) | [146](#) | [147](#) | [148](#) | [149](#) | [150](#) | [151](#) | [152](#) | [153](#) | [154](#) | [155](#) | [156](#) | [157](#) | [158](#) | [159](#) | [160](#) | [161](#) | [162](#) | [163](#) | [164](#) | [165](#) | [166](#) | [167](#) | [168](#) | [169](#) | [170](#) | [171](#) | [172](#) | [173](#) | [174](#) | [175](#) | [176](#) | [177](#) | [178](#) | [179](#) | [180](#) | [181](#) | [182](#) | [183](#) | [184](#) | [185](#) | [186](#) | [187](#) | [188](#) | [189](#) | [190](#) | [191](#) | [192](#) | [193](#) | [194](#) | [195](#) | [196](#) | [197](#) | [198](#) | [199](#) | [200](#) | [201](#) | [202](#) | [203](#) | [204](#) | [205](#) | [206](#) | [207](#) | [208](#) | [209](#) | [210](#) | [211](#) | [212](#) | [213](#) | [214](#) | [215](#) | [216](#) | [217](#) | [218](#) | [219](#) | [220](#) | [221](#) | [222](#) | [223](#) | [224](#) | [225](#) | [226](#) | [227](#) | [228](#) | [229](#) | [230](#) | [231](#) | [232](#) | [233](#) | [234](#) | [235](#) | [236](#) | [237](#) | [238](#) | [239](#) | [240](#) | [241](#) | [242](#) | [243](#) | [244](#) | [245](#) | [246](#) | [247](#) | [248](#) | [249](#) | [250](#) | [251](#) | [252](#) | [253](#) | [254](#) | [255](#) | [256](#) | [257](#) | [258](#) | [259](#) | [260](#) | [261](#) | [262](#) | [263](#) | [264](#) | [265](#) | [266](#) | [267](#) | [268](#) | [269](#) | [270](#) | [271](#) | [272](#) | [273](#) | [274](#) | [275](#) | [276](#) | [277](#) | [278](#) | [279](#) | [280](#) | [281](#) | [282](#) | [283](#) | [284](#) | [285](#) | [286](#) | [287](#) | [288](#) | [289](#) | [290](#) | [291](#) | [292](#) | [293](#) | [294](#) | [295](#) | [296](#) | [297](#) | [298](#) | [299](#) | [300](#) | [301](#) | [302](#) | [303](#) | [304](#) | [305](#) | [306](#) | [307](#) | [308](#) | [309](#) | [310](#) | [311](#) | [312](#) | [313](#) | [314](#) | [315](#) | [316](#) | [317](#) | [318](#) | [319](#) | [320](#) | [321](#) | [322](#) | [323](#) | [324](#) | [325](#) | [326](#) | [327](#) | [328](#) | [329](#) | [330](#) | [331](#) | [332](#) | [333](#) | [334](#) | [335](#) | [336](#) | [337](#) | [338](#) | [339](#) | [340](#) | [341](#) | [342](#) | [343](#) | [344](#) | [345](#) | [346](#) | [347](#) | [348](#) | [349](#) | [350](#) | [351](#) | [352](#) | [353](#) | [354](#) | [355](#) | [356](#) | [357](#) | [358](#) | [359](#) | [360](#) | [361](#) | [362](#) | [363](#) | [364](#) | [365](#) | [366](#) | [367](#) | [368](#) | [369](#) | [370](#) | [371](#) | [372](#) | [373](#) | [374](#) | [375](#) | [376](#) | [377](#) | [378](#) | [379](#) | [380](#) | [381](#) | [382](#) | [383](#) | [384](#) | [385](#) | [386](#) | [387](#) | [388](#) | [389](#) | [390](#) | [391](#) | [392](#) | [393](#) | [394](#) | [395](#) | [396](#) | [397](#) | [398](#) | [399](#) | [400](#) | [401](#) | [402](#) | [403](#) | [404](#) | [405](#) | [406](#) | [407](#) | [408](#) | [409](#) | [410](#) | [411](#) | [412](#) | [413](#) | [414](#) | [415](#) | [416](#) | [417](#) | [418](#) | [419](#) | [420](#) | [421](#) | [422](#) | [423](#) | [424](#) | [425](#) | [426](#) | [427](#) | [428](#) | [429](#) | [430](#) | [431](#) | [432](#) | [433](#) | [434](#) | [435](#) | [436](#) | [437](#) | [438](#) | [439](#) | [440](#) | [441](#) | [442](#) | [443](#) | [444](#) | [445](#) | [446](#) | [447](#) | [448](#) | [449](#) | [450](#) | [451](#) | [452](#) | [453](#) | [454](#) | [455](#) | [456](#) | [457](#) | [458](#) | [459](#) | [460](#) | [461](#) | [462](#) | [463](#) | [464](#) | [465](#) | [466](#) | [467](#) | [468](#) | [469](#) | [470](#) | [471](#) | [472](#) | [473](#) | [474](#) | [475](#) | [476](#) | [477](#) | [478](#) | [479](#) | [480](#) | [481](#) | [482](#) | [483](#) | [484](#) | [485](#) | [486](#) | [487](#) | [488](#) | [489](#) | [490](#) | [491](#) | [492](#) | [493](#) | [494](#) | [495](#) | [496](#) | [497](#) | [498](#) | [499](#) | [500](#) | [501](#) | [502](#) | [503](#) | [504](#) | [505](#) | [506](#) | [507](#) | [508](#) | [509](#) | [510](#) | [511](#) | [512](#) | [513](#) | [514](#) | [515](#) | [516](#) | [517](#) | [518](#) | [519](#) | [520](#) | [521](#) | [522](#) | [523](#) | [524](#) | [525](#) | [526](#) | [527](#) | [528](#) | [529](#) | [530](#) | [531](#) | [532](#) | [533](#) | [534](#) | [535](#) | [536](#) | [537](#) | [538](#) | [539](#) | [540](#) | [541](#) | [542](#) | [543](#) | [544](#) | [545](#) | [546](#) | [547](#) | [548](#) | [549](#) | [550](#) | [551](#) | [552](#) | [553](#) | [554](#) | [555](#) | [556](#) | [557](#) | [558](#) | [559](#) | [560](#) | [561](#) | [562](#) | [563](#) | [564](#) | [565](#) | [566](#) | [567](#) | [568](#) | [569](#) | [570](#) | [571](#) | [572](#) | [573](#) | [574](#) | [575](#) | [576](#) | [577](#) | [578](#) | [579](#) | [580](#) | [581](#) | [582](#) | [583](#) | [584](#) | [585](#) | [586](#) | [587](#) | [588](#) | [589](#) | [590](#) | [591](#) | [592](#) | [593](#) | [594](#) | [595](#) | [596](#) | [597](#) | [598](#) | [599](#) | [600](#) | [601](#) | [602](#) | [603](#) | [604](#) | [605](#) | [606](#) | [607](#) | [608](#) | [609](#) | [610](#) | [611](#) | [612](#) | [613](#) | [614](#) | [615](#) | [616](#) | [617](#) | [618](#) | [619](#) | [620](#) | [621](#) | [622](#) | [623](#) | [624](#) | [625](#) | [626](#) | [627](#) | [628](#) | [629](#) | [630](#) | [631](#) | [632](#) | [633](#) | [634](#) | [635](#) | [636](#) | [637](#) | [638](#) | [639](#) | [640](#) | [641](#) | [642](#) | [643](#) | [644](#) | [645](#) | [646](#) | [647](#) | [648](#) | [649](#) | [650](#) | [651](#) | [652](#) | [653](#) | [654](#) | [655](#) | [656](#) | [657](#) | [658](#) | [659](#) | [660](#) | [661](#) | [662](#) | [663](#) | [664](#) | [665](#) | [666](#) | [667](#) | [668](#) | [669](#) | [670](#) | [671](#) | [672](#) | [673](#) | [674](#) | [675](#) | [676](#) | [677](#) | [678](#) | [679](#) | [680](#) | [681](#) | [682](#) | [683](#) | [684](#) | [685](#) | [686](#) | [687](#) | [688](#) | [689](#) | [690](#) | [691](#) | [692](#) | [693](#) | [694](#) | [695](#) | [696](#) | [697](#) | [698](#) | [699](#) | [700](#) | [701](#) | [702](#) | [703](#) | [704](#) | [705](#) | [706](#) | [707](#) | [708](#) | [709](#) | [710](#) | [711](#) | [712](#) | [713](#) | [714](#) | [715](#) | [716](#) | [717](#) | [718](#) | [719](#) | [720](#) | [721](#) | [722](#) | [723](#) | [724](#) | [725](#) | [726](#) | [727](#) | [728](#) | [729](#) | [730](#) | [731](#) | [732](#) | [733](#) | [734](#) | [735](#) | [736](#) | [737](#) | [738](#) | [739](#) | [740](#) | [741](#) | [742](#) | [743](#) | [744](#) | [745](#) | [746](#) | [747](#) | [748](#) | [749](#) | [750](#) | [751](#) | [752](#) | [753](#) | [754](#) | [755](#) | [756](#) | [757](#) | [758](#) | [759](#) | [760](#) | [761](#) | [762](#) | [763](#) | [764](#) | [765](#) | [766](#) | [767](#) | [768](#) | [769](#) | [770](#) | [771](#) | [772](#) | [773](#) | [774](#) | [775](#) | [776](#) | [777](#) | [778](#) | [779](#) | [780](#) | [781](#) | [782](#) | [783](#) | [784](#) | [785](#) | [786](#) | [787](#) | [788](#) | [789](#) | [790](#) | [791](#) | [792](#) | [793](#) | [794](#) | [795](#) | [796](#) | [797](#) | [798](#) | [799](#) | [800](#) | [801](#) | [802](#) | [803](#) | [804](#) | [805](#) | [806](#) | [807](#) | [808](#) | [809](#) | [810](#) | [811](#) | [812](#) | [813](#) | [814](#) | [815](#) | [816](#) | [817](#) | [818](#) | [819](#) | [820](#) | [821](#) | [822](#) | [823](#) | [824](#) | [825](#) | [826](#) | [827](#) | [828](#) | [829](#) | [830](#) | [831](#) | [832](#) | [833](#) | [834](#) | [835](#) | [836](#) | [837](#) | [838](#) | [839](#) | [840](#) | [841](#) | [842](#) | [843](#) | [844](#) | [845](#) | [846](#) | [847](#) | [848](#) | [849](#) | [850](#) | [851](#) | [852](#) | [853](#) | [854](#) | [855](#) | [856](#) | [857](#) | [858](#) | [859](#) | [860](#) | [861](#) | [862](#) | [863](#) | [864](#) | [865](#) | [866](#) | [867](#) | [868](#) | [869](#) | [870](#) | [871](#) | [872](#) | [873](#) | [874](#) | [875](#) | [876](#) | [877](#) | [878](#) | [879](#) | [880](#) | [881](#) | [882](#) | [883](#) | [884](#) | [885](#) | [886](#) | [887](#) | [888](#) | [889](#) | [890](#) | [891](#) | [892](#) | [893](#) | [894](#) | [895](#) | [896](#) | [897](#) | [898](#) | [899](#) | [900](#) | [901](#) | [902](#) | [903](#) | [904](#) | [905](#) | [906](#) | [907](#) | [908](#) | [909](#) | [910](#) | [911](#) | [912](#) | [913](#) | [914](#) | [915](#) | [916](#) | [917](#) | [918](#) | [919](#) | [920](#) | [921](#) | [922](#) | [923](#) | [924](#) | [925](#) | [926](#) | [927](#) | [928](#) | [929](#) | [930](#) | [931](#) | [932](#) | [933](#) | [934](#) | [935](#) | [936](#) | [937](#) | [938](#) | [939](#) | [940](#) | [941](#) | [942](#) | [943](#) | [944](#) | [945](#) | [946](#) | [947](#) | [948](#) | [949](#) | [950](#) | [951](#) | [952](#) | [953](#) | [954](#) | [955](#) | [956](#) | [957](#) | [958](#) | [959](#) | [960](#) | [961](#) | [962](#) | [963](#) | [964](#) | [965](#) | [966](#) | [967](#) | [968](#) | [969](#) | [970](#) | [971](#) | [972](#) | [973](#) | [974](#) | [975](#) | [976](#) | [977](#) | [978](#) | [979](#) | [980](#) | [981](#) | [982](#) | [983](#) | [984](#) | [985](#) | [986](#) | [987](#) | [988](#) | [989](#) | [990](#) | [991](#) | [992](#) | [993](#) | [994](#) | [995](#) | [996](#) | [997](#) | [998](#) | [999](#) | [1000](#)

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

Crawler properties

Name yt-json-crawler	IAM role yt-glue-role	Database yt-raw-db	State READY
Description -	Security configuration -	Lake Formation configuration -	Table prefix -
Maximum table threshold -			

Advanced settings

Crawler runs | Schedule | Data sources | Classifiers | Tags

Crawler runs (1)

The list of crawler runs for this crawler.

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
May 10, 2024 at 07:20:14	May 10, 2024 at 07:21:13	59 s	Completed	-	1 table change, 0 partition changes

10. Review the Table and Perform the Query analysis

- The Crawler will create a table with the json data in the specified database location

raw_json

Last updated (UTC)
May 10, 2024 at 07:23:03

Version 0 (Current version) | Actions

Table overview | **Data quality** New

Table details | Advanced properties

Name raw_json	Description -	Database yt-raw-db	Classification JSON
Location s3://yt-initial/yt/raw-json/	Connection -	Deprecated -	Last updated May 10, 2024 at 07:21:13
Input format org.apache.hadoop.mapred.TextInputFormat	Output format org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat	Serde serialization lib org.openx.data.jsonserde.JsonSerDe	

Schema | Partitions | Indexes | Column statistics - new

Schema (3)

Edit schema as JSON | Edit schema

#	Column name	Data type	Partition key	Comment
1	kind	string	-	-
2	etag	string	-	-
3	items	array	-	-

- 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 extraction of specified data.

The screenshot shows the AWS Glue Data Catalog Query Editor interface. In the top navigation bar, there are tabs for 'Query 2', 'Query 3', and 'Query 4'. The 'Query 4' tab is active, showing the SQL query: `SELECT * FROM "AwsDataCatalog"."yt-raw-db"."raw_json" limit 10;`. Below the query editor, the 'Tables and views' section lists 'Tables (1)' containing 'raw_json' and 'Views (0)'. The main panel displays the query results, which failed with the error message: `HIVE_CURSOR_ERROR: Row is not a valid JSON Object - JSONException: A JSONObject text must end with '}' at 2 [character 3 line 1]`. The status bar at the bottom indicates a run time of 214 ms.

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)

The screenshot shows the 'Create New Function' configuration page. The 'Function name' field is set to 'yt-json-par-lambda'. The 'Runtime' is selected as 'Python 3.8'. Under 'Architecture', 'x86_64' is chosen. In the 'Permissions' section, it is noted that Lambda will create an execution role. The 'Change default execution role' section shows 'Execution role' with the option 'Use an existing role' selected, and 'yt-lambda-role' is chosen from the dropdown. The 'Existing role' note states that the role must have permission to upload logs to Amazon CloudWatch Logs.

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

AWS Glue > Databases > yt-cleaned-db

yt-cleaned-db

Last updated (UTC) May 10, 2024 at 07:34:30 Edit Delete

Database properties

Name yt-cleaned-db	Description -	Location -	Created on (UTC) May 10, 2024 at 07:34:25
-----------------------	------------------	---------------	--

Tables (0)

Last updated (UTC) May 10, 2024 at 07:34:32 Edit Add tables using crawler Add table

View and manage all available tables.

Filter tables < 1 > ⚙

Name	Database	Location	Classification	Deprecated	View data	Data quality
No available tables						

- 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

Amazon S3 > Buckets > yt-clean

yt-clean

Info

Objects (0) Info

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix < 1 > ⚙

Name	Type	Last modified	Size	Storage class
No objects You don't have any objects in this bucket.				

Upload

- 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.

File Edit Find View Go Tools Window Test Deploy

lambda_function x Environment Var x Execution results x

Go to Anything (Ctrl-P)

Environment

```

1 import awswrangler as wr
2 import pandas as pd
3 import urllib.parse
4 import os
5
6 os_input_s3_cleansed_layer = os.environ['s3_cleansed_layer']
7 os_input_glue_catalog_db_name = os.environ['glue_catalog_db_name']
8 os_input_glue_catalog_table_name = os.environ['glue_catalog_table_name']
9 os_input_write_data_operation = os.environ['write_data_operation']
10
11
12 def lambda_handler(event, context):
13     # Get the object from the event and show its content type
14     bucket = event['Records'][0]['s3']['bucket']['name']
15     key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'], encoding='utf-8')
16     try:
17         # Creating DF from content
18         df_raw = wr.s3.read_json('s3://{}{}'.format(bucket, key))
19
20         # Extract required columns:
21         df_step_1 = pd.json_normalize(df_raw['items'])
22
23         # Write to S3
24         wr_response = wr.s3.to_parquet(
25             df=df_step_1,
26             path=os_input_s3_cleansed_layer,
27             dataset=True,
28             database=os_input_glue_catalog_db_name,
29             table=os_input_glue_catalog_table_name,
30             mode=os_input_write_data_operation
31         )
32     
```

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

Timeout
5 min 0 sec

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

Use an existing role
 Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

yt-lambda-role ▼ C

[View the yt-lambda-role role](#) ▼ on the IAM console.

[Cancel](#) [Save](#)

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

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#)

Key	Value	Remove
glue_catalog_db_name	yt-cleaned-db	Remove
glue_catalog_table_name	cleaned-json	Remove
s3_cleaned_layer	yt-clean	Remove
write_data_operation	append	Remove

[Add environment variable](#)

▶ [Encryption configuration](#)

[Cancel](#) [Save](#)

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

AWS

Home > Applications > Application details

aws-data-wrangler-layer-py3-8

arn:aws:serverlessrepo:us-east-1:336392948345:applications/aws-data-wrangler-layer-py3-8

AWS Data Wrangler <https://aws-data-wrangler.readthedocs.io/en/stable/>

Deploy 654 deployments

AWS Data Wrangler Lambda Layer (Python 3.8) [DEPRECATED]

[Readme](#) | [License](#) | [Permissions](#)

AWS Data Wrangler

Choose a layer

Layer source [Info](#)
Choose from layers with a compatible runtime and instruction set architecture or specify the Amazon Resource Name (ARN) of a layer version. You can also [create a new layer](#).

AWS layers
Choose a layer from a list of layers provided by AWS.

Custom layers
Choose a layer from a list of layers created by your AWS account or organization.

Specify an ARN
Specify a layer by providing the ARN.

Custom layers
Layers created by your AWS account or organization that are compatible with your function's runtime.
AWSDataWrangler-Python38 ▾

Version
1 ▾

Cancel **Add**

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

Template - optional

s3-put

Event JSON

Format JSON

```
4     "eventVersion": "2.0",
5     "eventSource": "aws:s3",
6     "awsRegion": "us-east-1",
7     "eventTime": "1970-01-01T00:00:00.000Z",
8     "eventName": "ObjectCreated:Put",
9     "userIdentity": {
10         "principalId": "EXAMPLE"
11     },
12     "requestParameters": {
13         "sourceIPAddress": "127.0.0.1"
14     },
15     "responseElements": {
16         "x-amz-request-id": "EXAMPLE123456789",
17         "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGHI"
18     },
19     "s3": {
20         "s3SchemaVersion": "1.0",
21         "configurationId": "testConfigRule",
22         "bucket": {
23             "name": "yt-initial",
24             "ownerIdentity": {
25                 "principalId": "EXAMPLE"
26             },
27             "arn": "arn:aws:s3:::yt-initial"
28         },
29         "object": {
30             "key": "yt/raw-json/CA_category_id.json",
31             "size": 1024,
32             "eTag": "0123456789abcdef0123456789abcdef",
33             "sequencer": "0A1B2C3D4E5F678901"
34         }
35     }
```

Cancel

Invoke

Save

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

The screenshot shows the AWS Lambda console interface. At the top, a green banner displays the message "Successfully updated the function yt-json-par-lambda.". Below the banner, the function name "yt-json-par-lambda" is visible. The "Code source" tab is selected, showing the code structure under "lambda_function". The "Test" tab is also present. In the center, the "Execution result" tab is open, displaying the following JSON output:

```
{  "path": [    "s3://vt-clean/13545bz18d16483882d0f8fd0c84df288.snappy.parquet"  ],  "partitions_values": ()}
```

Below the output, the "Function Logs" section shows the execution details:

```
START RequestId: 9c3b4b2a-6acb-4a62-bc8c-b0fe993800ca Version: $LATEST
END RequestId: 9c3b4b2a-6acb-4a62-bc8c-b0fe993800ca
REPORT RequestId: 9c3b4b2a-6acb-4a62-bc8c-b0fe993800ca Duration: 10664.75 ms Billed Duration: 10665 ms Memory Size: 128 MB Max Memory Used: 128 MB Init Duration: 3793.30 ms
Request ID
9c3b4b2a-6acb-4a62-bc8c-b0fe993800ca
```

- Check for the Confirmation of Parquet file generation

Amazon S3 > Buckets > yt-clean

yt-clean [Info](#)

Objects (1) [Info](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

[Search](#)

Name	Type	Last modified	Size	Storage class
13545b218d16483882d8f8d0c8_4df280.snappy.parquet	parquet	May 10, 2024, 14:35:56 (UTC+05:30)	6.1 KB	Standard

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

AWS Glue > Tables > json_cleaned

json_cleaned

Table overview Data quality [New](#)

Table details Advanced properties

Name: json_cleaned	Description:	Database: db_cleaned	Classification: Parquet
Location: s3://yt-cleaned	Connector:	Deprecated:	Last updated: May 10, 2024 at 09:07:43
Input format: org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat	Output format: org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat	Serde serialization lib: org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveClientSerDe	Last updated: May 10, 2024 at 09:06:24

- 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.

i Athena now supports typeahead code suggestions to speed up SQL query development

Typeahead suggestions are turned on by default. You can change this setting in query editor preferences.

Data [C](#) <

Query 2 : X | Query 3 : X | Query 4 : X | **Query 5 : X**

1 `SELECT * FROM "AwsDataCatalog"."yt-cleaned-db"."cleaned_json" limit 10;`

Data source: AwsDataCatalog

Database: yt-cleaned-db

Tables and views [Create](#) [⚙️](#)

Tables (1) < 1 >

+ cleaned_json : [View](#) [Edit](#) [Delete](#)

Views (0) < 1 >

SQL Ln 1, Col 1

[Run again](#) [Explain](#) [Cancel](#) [Clear](#) [Create](#)

Query results		Query stats			
Completed		Time in queue: 63 ms Run time: 328 ms Data scanned: 2.11 KB			
Results (10)		Copy Download results			
<input type="text"/> Search rows		< 1 > Copy			
#	kind	etag	id	snippet_channelid	snippet_title
1	youtube#videoCategory	"ld9biNPKjAjjV7EZ4EKeGhrao/Xy1mB4_yLrHy_BmKmPBggty2mZQ"	1	UCBR8-60-B28hp2BmDPdntcQ	Film & Animation
2	youtube#videoCategory	"ld9biNPKjAjjV7EZ4EKeGhrao/UZ1oIiLz2dxlhO45ZTFR3a3NyTA"	2	UCBR8-60-B28hp2BmDPdntcQ	Autos & Vehicles
3	youtube#videoCategory	"ld9biNPKjAjjV7EZ4EKeGhrao/nqRiq97-xe5XRZTxkbnKFVe5Lmg"	10	UCBR8-60-B28hp2BmDPdntcQ	Music
4	youtube#videoCategory	"ld9biNPKjAjjV7EZ4EKeGhrao/HwXKamM1Q20q9BN-oBJavSGkfDI"	15	UCBR8-60-B28hp2BmDPdntcQ	Pets & Animals
5	youtube#videoCategory	"ld9biNPKjAjjV7EZ4EKeGhrao/9GQMSRjrZdHeb10EM1XVQ9zbGec"	17	UCBR8-60-B28hp2BmDPdntcQ	Sports
6	youtube#videoCategory	"ld9biNPKjAjjV7EZ4EKeGhrao/FJwVpGCVZ1yiJrqZbpqe68Sy_OE"	18	UCBR8-60-B28hp2BmDPdntcQ	Short Movies
7	youtube#videoCategory	"ld9biNPKjAjjV7EZ4EKeGhrao/M-3iD9dwK7YJCafRf_DkLN8CouA"	19	UCBR8-60-B28hp2BmDPdntcQ	Travel & Events
8	youtube#videoCategory	"ld9biNPKjAjjV7EZ4EKeGhrao/WmA0qYEfjWsAoyJFSw2zinhn2wM"	20	UCBR8-60-B28hp2BmDPdntcQ	Gaming
9	youtube#videoCategory	"ld9biNPKjAjjV7EZ4EKeGhrao/EapFaGYG7K0StIXf8aba249tdM"	21	UCBR8-60-B28hp2BmDPdntcQ	Videoblogging
10	youtube#videoCategory	"ld9biNPKjAjjV7EZ4EKeGhrao/xld8RX7vRN8rqkbYZbNlytUQDro"	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

Set crawler properties

Name yt-csv-crawler	Description	Tags
------------------------	-------------	------

Step 2: Choose data sources and classifiers

Data sources (1) Info
The list of data sources to be scanned by the crawler.

Type S3	Data source s3://yt-initial/yt/raw-csv/	Parameters Recrawl all
------------	--	---------------------------

Step 3: Configure security settings

Configure security settings

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

Step 4: Set output and scheduling

Set output and scheduling

Database yt+raw-db	Table prefix - optional	Maximum table threshold - optional	Schedule 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.				
<input type="text"/> Filter data	<input type="text"/> Filter by a date and time range	Stop run	View CloudWatch logs	View run details
Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours
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.

The screenshot shows two pages from the AWS Glue console:

- Table details:** Shows basic metadata for the table: Name (raw_csv), Description (-), Database (yt-raw-db), Classification (CSV). Location is s3://yt-initial/yt/raw-csv/. Input format is org.apache.hadoop.mapred.TextInputFormat and Output format is org.apache.hadoop.hive.io.HiveIgnoreKeyTextOutputFormat. Serde serialization lib is org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe.
- Schema (17):** Shows the table schema with 7 columns: video_id, trending_date, title, channel_title, category_id, publish_time, and tags. All columns are string type and not marked as partition keys or comments.

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.

The screenshot shows the AWS Athena Query Editor interface:

- Data:** Data source is AwsDataCatalog, Database is yt-cleaned-db.
- Tables and views:** Tables (1) include cleaned_json. Views (0) are listed.
- Query 7:** The current query is:


```
1 SELECT * FROM "AwsDataCatalog"."yt-raw-db"."raw_csv" as r
2 join "yt-cleaned-db"."cleaned_json" as c
3 on r.category_id = cast(c.id as int)
4 limit 10
```
- Buttons at the bottom:** Run again, Explain, Cancel, Clear, Create.

#	video_id	trending_date	title	channel_title
1	Kf2-86o5S1o	18.05.03	"Film Theory: The Bee Movie LIED To You!"	"The Film Theorists"
2	Juad74hE6rs	18.05.03	"Crazy Frosting Recipe: The Best Buttercream Frosting with Endless Flavor Variations!"	"Gemma Stafford"
3	MgMLdq9DnNc	18.05.03	"February Favorites 2018"	"Jenn Im"
4	xwXP7vB4mlY	18.05.03	"She Ruined The Surprise Gender Reveal"	"KKandbabyJ"
5	9bAiXJoNdyO	18.05.03	"One thing that makes you a better friend"	"Anna Akana"
6	n9JVbRBqfY	18.05.03	"LEGENDARY All You Can Eat Buffet in Manila Philippines - Spiral Buffet Review"	"Strictly Dumpling"
7	n_S8d_1KVhU	18.05.03	"How Satellites Capture 400 Megapixel Images Of Earth's Globe - Himawari 8 & GOES-16"	"Scott Manley"

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

Schema (1/6)							Delete	Edit	Add
View and manage the table schema.									
	#	Column name	Data type	Partition key	Comment				
<input type="checkbox"/>	1	kind	string	-	-				
<input type="checkbox"/>	2	etag	string	-	-				
<input checked="" type="checkbox"/>	3	id	bigint	-	-				
<input type="checkbox"/>	4	snippet_channelid	string	-	-				
<input type="checkbox"/>	5	snippet_title	string	-	-				
<input type="checkbox"/>	6	snippet_assigna...	boolean	-	-				

Cancel Save as new table version

- 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.

The screenshot shows the AWS Glue Data Catalog Query Editor interface. At the top, there are tabs for 'Query 2' through 'Query 7'. The 'Query 7' tab is active, displaying the following SQL code:

```

1 SELECT * FROM "AwsDataCatalog"."yt-cleaned-db"."raw_csv" as r
2 join "yt-cleaned-db"."cleaned_json" as c
3 on r.category_id = c.id
4 limit 10

```

Below the code, it says 'SQL Ln 3, Col 25'. There are buttons for 'Run again', 'Explain', 'Cancel', 'Clear', and 'Create'. A 'Reuse query results' checkbox is checked, indicating results are valid up to 60 minutes ago.

The 'Query results' tab is selected, showing a red box around the error message: 'Failed' with a red circle icon. The message reads:

HIVE_BAD_DATA: Field id's type BINARY in parquet file s3://yt-clean/yt/555cd0d1cd264696850f9bca1cd8af14.snappy.parquet is incompatible with type bigint defined in table schema

This query ran against the "yt-cleaned-db" database, unless qualified by the query. Please post the error message on our [forum](#) or contact [customer support](#) with Query Id: 30844e07-5a7d-4f85-8cad-8164cc18c2ee

- 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.

The screenshot shows the AWS Lambda function configuration page for 'yt-json-par-lambda'. The 'Code source' tab is selected, showing the function code in 'lambda_function.py'.

```

# This is a sample Python script for a Lambda function.

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }

```

The 'Test' tab is selected, showing the execution result. The status is 'Succeeded' with a duration of 10664.75 ms. The response body is:

```

{
  "paths": [
    "s3://yt-clean/13545b218d16482d8f8d0c84df280.snappy.parquet"
  ],
  "partitions_values": {}
}

```

The 'Function Logs' section shows the request and response details.

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

The screenshot shows the AWS Glue Data Catalog Query Editor interface. The 'Data' tab is selected, showing the 'Tables and views' section with 'json_cleaned' listed. The 'Tables' section shows columns: kind, msg, id, import_channelid, import_file, and import_compatible.

The 'Query 7' tab is active, displaying the same SQL code as before:

```

1 SELECT * FROM "AwsDataCatalog"."yt-cleaned-db"."raw_csv" as r
2 join "yt-cleaned-db"."cleaned_json" as c
3 on r.category_id = c.id
4 limit 10

```

Below the code, it says 'SQL Ln 3, Col 22'. There are buttons for 'Run again', 'Explain', 'Cancel', 'Clear', and 'Create'. A 'Reuse query results' checkbox is checked, indicating results are valid up to 60 minutes ago.

The 'Query results' tab is selected, showing a green box around the message: 'Completed' with a green circle icon. The message reads:

Time in queue: 55 ms Run time: 1.216 sec Data scanned: 10.06 MB

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

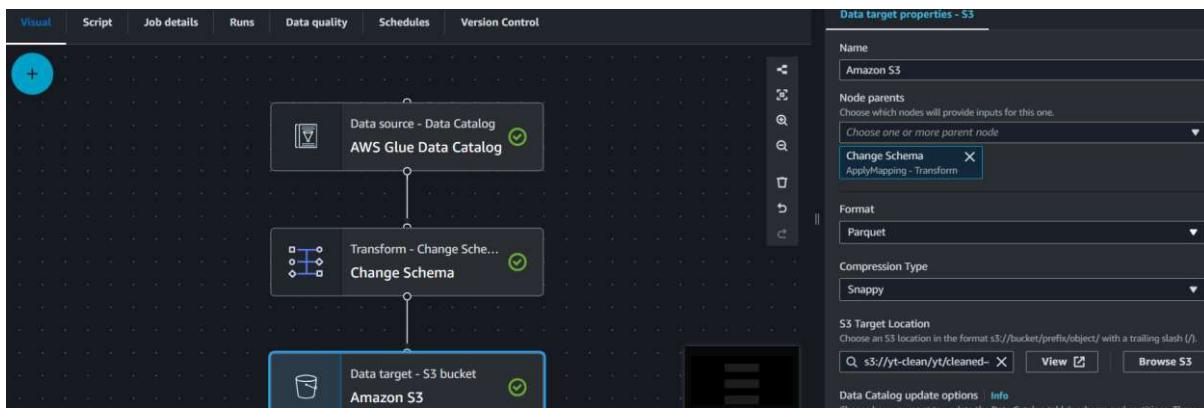
The screenshot shows the AWS Glue Visual ETL interface. At the top, there are tabs for 'Visual', 'Script', 'Job details', 'Runs', 'Data quality', 'Schedules', and 'Version Control'. The 'Visual' tab is selected. In the main workspace, a 'Data source - Data Catalog' node (labeled 'AWS Glue Data Catalog') is connected to a 'Data preview' node. The 'Data preview' node shows 200 rows of data from a 'raw_csv' table in the 'yt-raw-db' database. The columns include video_id, trending_date, title, channel_title, category_id, and publish_time. A preview of one row shows: video_id RYs0BkX3lh4, trending_date 17.14.11, title SECRETS REVEALED! HO WI LAY MY LACE WIGS!, channel_title MsAaliyahJay, category_id 26, and publish_time 2022-02-02.

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

The screenshot shows the AWS Glue Visual ETL interface. The main workspace now includes a 'Transform' node and a 'Change Schema' node. The 'Transform' node is connected to the 'Data source - Data Catalog' node, which is then connected to the 'Change Schema' node. The 'Change Schema' node is connected to the 'Data preview' node. The 'Change Schema' node has a table titled 'Change Schema (Apply mapping)' showing the transformation rules:

Source key	Target key	Data type	Drop
video_id	video_id	string	<input type="checkbox"/>
trending_date	trending_dat	string	<input type="checkbox"/>
title	title	string	<input type="checkbox"/>
channel_title	channel_title	string	<input type="checkbox"/>
category_id	category_id	bigint	<input type="checkbox"/>
publish_time	publish_time	string	<input type="checkbox"/>
tags	tags	string	<input type="checkbox"/>
views	views	bigint	<input type="checkbox"/>
likes	likes	bigint	<input type="checkbox"/>

- 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



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 ▼ C

▶ 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 ▼ X

Add a partition key

yt-csv-par-job Actions ▾ Save Run

Visual Script Job details Runs Data quality Schedules Version Control

Script (Locked) Info

```

1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7
8 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
9 sc = SparkContext()
10 glueContext = GlueContext(sc)
11 spark = glueContext.spark_session
12 job = Job(glueContext)
13 job.init(args['JOB_NAME'], args)
14
15 # Script generated for node AWS Glue Data Catalog
16 AWSGlueDataCatalog_node1715341147410 = glueContext.create_dynamic_frame.from_catalog(database="yt-raw-db", table_name="raw_csv", transformation_ctx="AWSGlueDataCatalog_node1715341147410")
17
18 # Script generated for node Change Schema
19 ChangeSchema_node1715341212976 = ApplyMapping.apply(frame=AWSGlueDataCatalog_node1715341147410, mappings=[{"video_id": "string", "string": "video_id", "video_id": "string"}, {"trending_date": "string", "string": "trending_date", "trending_date": "string"}])
20
21 # Script generated for node Amazon S3
22 AmazonS3_node1715341258708 = glueContext.getSink(path="s3://yt-clean/yt/cleaned-csv/", connection_type="s3", updateBehavior="UPDATE_IN_DATABASE", partitionKeys=["partition_0"], enableUpdateCatalog=True, t
23 AmazonS3_node1715341258708.setCatalogInfo(catalogDatabase="yt-cleaned-db", catalogTableName="clean-csv")
24 AmazonS3_node1715341258708.setFormat("glueparquet", compression="snappy")
25 AmazonS3_node1715341258708.writeFrame(ChangeSchema_node1715341212976)
26 job.commit()
  
```

16. Run the ETL job.

- It creates the parquets file in the s3 and tables in the cleaned database

The screenshot shows the AWS Glue Job Runner interface for the job 'yt-csv-par-job'. The 'Runs' tab is selected, displaying one run that has succeeded. The run details include: Start time (Local) - 05/10/2024 17:14:25, End time (Local) - 05/10/2024 17:15:42, Duration - 1 m 7 s, Capacity (DPUs) - 10 DPUs, Worker type - G.1X, and Glue version - 4.0. The status is 'Succeeded' with 0 retries. The interface also includes tabs for 'Visual', 'Script', 'Job details', 'Data quality', 'Schedules', and 'Version Control', along with 'Actions', 'Save', and 'Run' buttons.

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

The screenshot shows the AWS S3 Objects list for a bucket. There are three objects listed under the 'Name' column: 'partition_0=CA-csv/' (Folder), 'partition_0=GB-csv/' (Folder), and 'partition_0=US-csv/' (Folder). The 'Actions' menu is visible above the list, and a search bar at the top allows filtering by prefix.

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

The screenshot shows the AWS Glue Crawler properties for 'wrapper-around-parquet-catalog'. The crawler was last updated on May 9, 2024 at 16:21:39. The 'Crawler properties' section displays the following configuration:

Name	IAM role	Database	State
wrapper-around-parquet-catalog	glue-s3-full-access	cleaned-csv-database	READY
Description	Security configuration	Lake Formation configuration	Table prefix
Maximum table threshold	-	-	-

A 'Advanced settings' button is located at the bottom left of the properties section.

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

The screenshot shows the 'Crawler runs' section of the AWS Glue console. It displays a single crawler run with the following details:

- Crawler runs (1)**
- Start time (UTC)**: May 9, 2024 at 16:22:26
- End time (UTC)**: May 9, 2024 at 16:23:19
- Current/last duration**: 52 s
- Status**: Completed
- DPU hours**: -

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

The screenshot shows the 'Databases' section of the AWS Glue console. It displays the 'statistics-cleaned' database with the following properties:

Name	Description	Location	Created on (UTC)
statistics-cleaned	-	-	May 10, 2024 at 05:07:47

The 'Tables' section shows two tables:

Name	Database	Location	Classification	Deprecation	Action
cleaned_csv_dat	statistics-cleaned	s3://s3-cleaned-	Parquet	-	Table data
crawl_processed	statistics-cleaned	s3://s3-cleaned-	Parquet	-	Table data

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

The screenshot shows the AWS Athena Query Editor interface. A query is being run against the 'cleaned_csv_dat' table in the 'yt-cleaned-db' database:

```
1 SELECT * FROM "AwsDataCatalog"."yt-cleaned-db"."clean-csv" limit 10;
```

The results pane shows the following output:

Column 1	Column 2	Column 3
Region	Country	City
North America	USA	New York
North America	USA	Los Angeles
Europe	UK	London
Europe	UK	Glasgow
Asia	China	Beijing
Asia	China	Shanghai
South America	Brazil	Sao Paulo
South America	Brazil	Rio de Janeiro

Query results		Query stats	
Completed		Time in queue: 53 ms	Run time: 894 ms
		Data scanned: 128.89 KB	
Results (10)		Copy Download results	
#	video_id	trending_date	title
1	rHwDegptbl4	17.14.11	Dashcam captures truck's near miss with child in Norway
2	J_QGZspO4gg	17.14.11	Sia - Snowman
3	RYs08kX3lh4	17.14.11	SECRETS REVEALED! HOW I LAY MY LACE WIGS! AALIYAHJAY
4	4FDpjKdlIxA	17.14.11	Waking Up with Sam Harris #103 - American Fantasies (with Kurt Andersen)
5	kGOmPmlLndU	17.14.11	The reputation Secret Sessions
6	_wM_jY_rass	17.14.11	Bone on Labour HQ
7	Jj0uBQ7j5c4	17.14.11	Ex engineer leaks how marketing works in the bike industry
8	WqMOHepSpkU	17.14.11	TYSON FURY TO ANTHONY JOSHUA - YOU AINT SEEN **** LIKE THIS YOU WEIGHT-LIFTER!- w/ CHISORA & BUFFER
9	UNTOl5RzCek	17.14.11	Kyle Hits Cartman REAL HARD!!!
10	S0A4hBJHND4	17.14.11	Svjetska ekskluziva: Provala u stan Dejana Lovrena

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.

Trigger configuration Info

S3 aws asynchronous storage ▾

Bucket
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.

X C

Bucket region: ap-south-1

Event types
Select the events that you want to trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

All object create events X ▾

Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.

yt/raw-json/

Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.

.json

Recursive invocation
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. [Learn more](#)

I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

- deleting json files

Successfully deleted objects
View details below.

Delete objects: status

The information below will no longer be available after you navigate away from this page.

Summary

Source	Successfully deleted	Failed to delete
s3://yt-clean/yt/	1 object, 6.1 KB	0 objects

- Reupload the json files

aws Services Search [Alt+S] Mumbai ▾ role-IIHT-LAB/lab-session

☰ **Upload succeeded**
View details below.

Files and folders (6 Total, 171.7 MB)

Find by name

Name	Folder	Type	Size	Status	Error
CA_categor...	yt/raw-json/	application/...	7.7 KB	✓ Succeeded	-
GB_categor...	yt/raw-json/	application/...	8.0 KB	✓ Succeeded	-
US_categor...	yt/raw-json/	application/...	8.3 KB	✓ Succeeded	-

- Confirmation for the parquet files generated

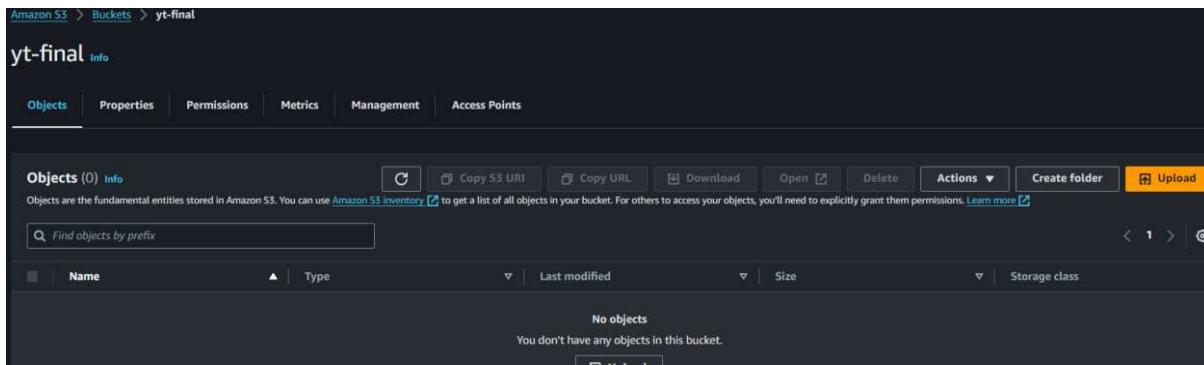
Objects (4) Info

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

Name	Type	Last modified	Size	Storage class
0fe22480eafa4deba6db164af4dd7fea.s...	parquet	May 10, 2024, 17:55:43 (UTC+05:30)	6.1 KB	Standard
c59ac3530a1c44ed85ba52f76f8fa499.s...	parquet	May 10, 2024, 17:55:42 (UTC+05:30)	6.1 KB	Standard
cleaned-csv/	Folder	-	-	-
f43b7eea79014a849c935c7773a1ea50.s...	parquet	May 10, 2024, 17:55:41 (UTC+05:30)	6.1 KB	Standard

18. Create Final S3 Bucket for the Analytical purpose.

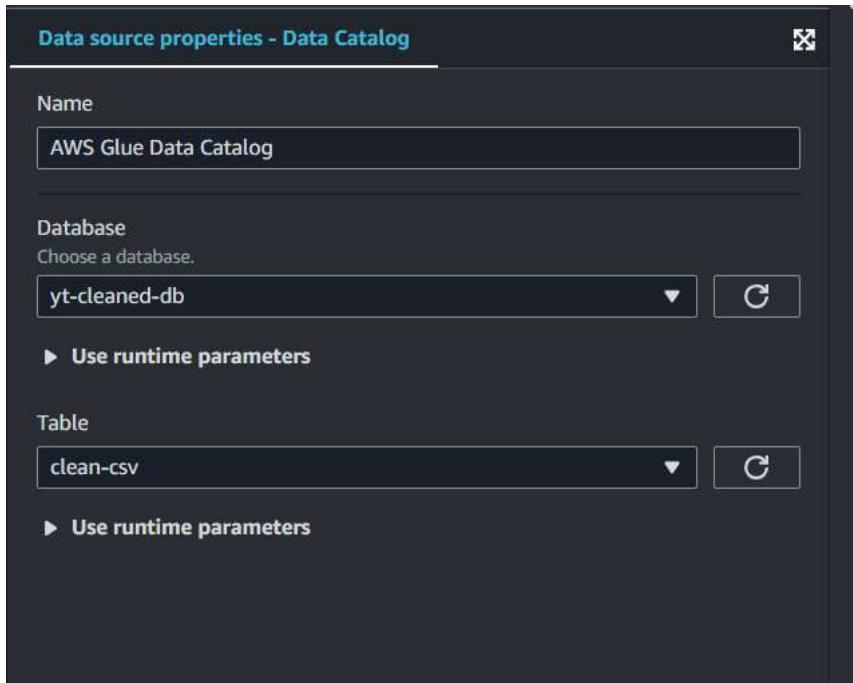


19. Create a ETL job combining the both json and csv Data catalogs

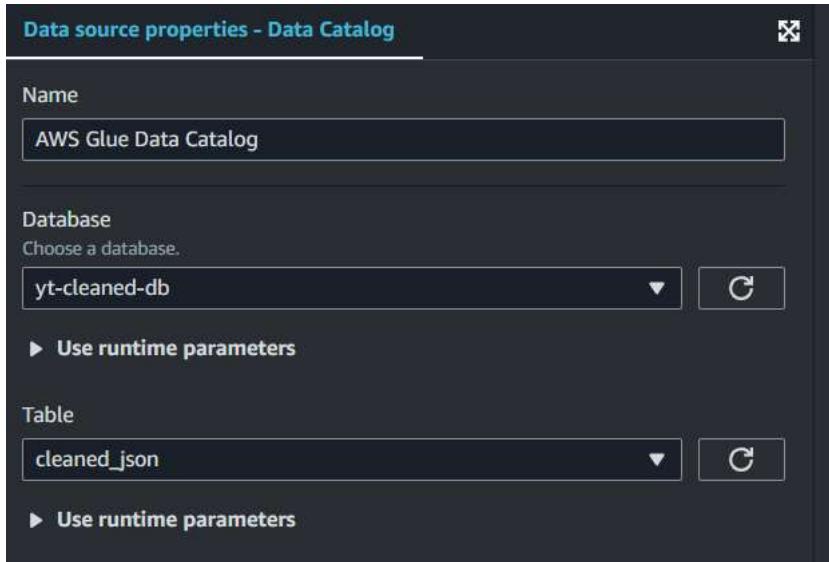
- We create the Glue ETL job for combining the csv and json Data Catalog and store it in a new table, so that we can perform data visualization on the that single table using some visualisation tools.
- We use Visual ETL for designing the job flow



- 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.

Transform

Name
Join

Node parents
Choose which nodes will provide inputs for this one.
Choose one or more parent node

AWS Glue Data Catalog X
Catalog - DataSource

AWS Glue Data Catalog X
Catalog - DataSource

Join type
Select the type of join to perform.

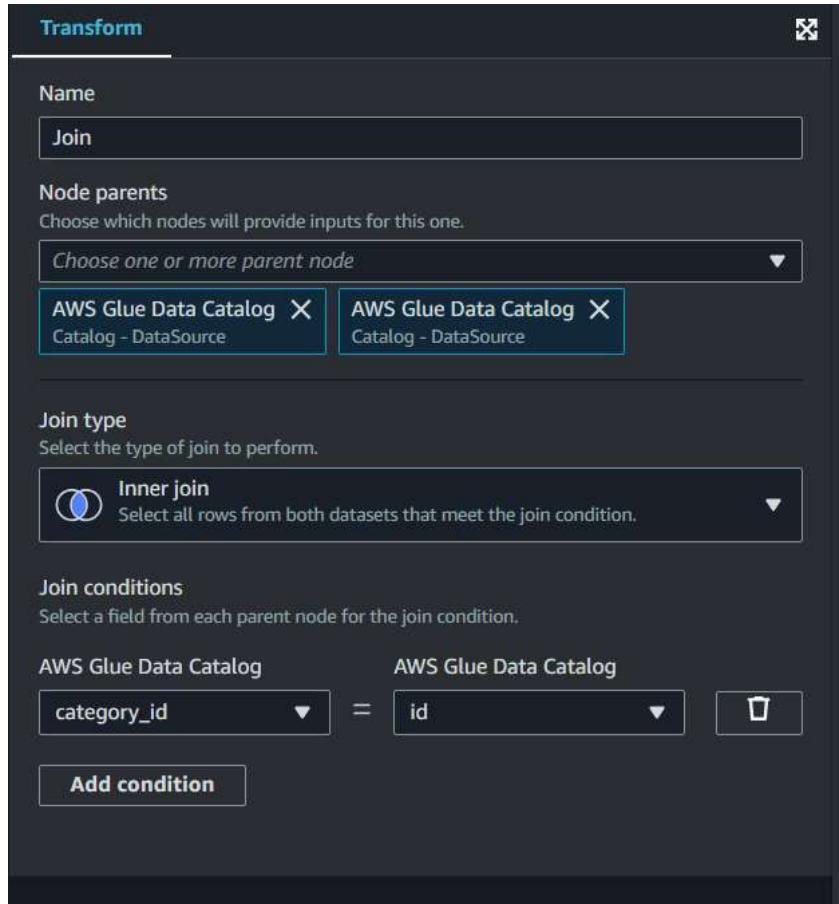
Inner join
Select all rows from both datasets that meet the join condition.

Join conditions
Select a field from each parent node for the join condition.

AWS Glue Data Catalog AWS Glue Data Catalog

category_id ▼ = id ▼ Delete

Add condition



This screenshot shows the configuration of a 'Join' transformation in the AWS Glue Data Catalog. The 'Name' is set to 'Join'. Under 'Node parents', two datasets are selected: 'AWS Glue Data Catalog - Catalog - DataSource' and 'AWS Glue Data Catalog - Catalog - DataSource'. The 'Join type' is set to 'Inner join'. In the 'Join conditions' section, a condition 'category_id = id' is defined between the two datasets. There is also a 'Add condition' button available.

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

Data target properties - S3

Name
Amazon S3

Node parents
Choose which nodes will provide inputs for this one.
Choose one or more parent node ▾

Join X
Join - 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-final X View Browse S3

After you save your job, it will use Glue Studio's optimized Parquet writer. X

- 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.

Data target properties - S3

Data Catalog update options [Info](#)

Choose how you want to update the Data Catalog table's schema and partitions. These options will only apply if the Data Catalog table is an S3 backed source.

- 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.

Choose one database

▲
C
X

Q db_analytics

▲
C
X

No databases available

Table name

Enter a table name for the AWS Glue Data Catalog.

analytics_table

Partition keys - optional

Add partition keys.

Partition (0)

partition_0

▼

Delete

Partition (1)

category_id

▼

Delete

20. Run the ETL job

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

yt-csv-join-json-job								Last modified on 5/10/2024, 6:13:30 PM	Actions	Save	Run
Visual	Script	Job details	Runs	Data quality	Schedules	Version Control					
Job runs (1/1) Info											
Last updated (UTC)	C	View details	Stop job run	Table View	Card View						
May 10, 2024 at 12:45:07							<	1	>		
<input type="checkbox"/> Filter job runs by property											
Run status	Retries	Start time (Local)	End time (Local)	Duration	Capacity	Worker type	Glue version				
Succeeded	0	05/10/2024 18:13:34	05/10/2024 18:14:55	1 m 12 s	10 DPU	G.1X	4.0				

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

The screenshot shows the Amazon S3 console with the 'yt-final' bucket selected. The 'Objects' tab is active, displaying three items: 'partition_0=CA-csv/' (Folder), 'partition_0=GB-csv/' (Folder), and 'partition_0=US-csv/' (Folder). The interface includes standard S3 actions like Copy S3 URI, Copy URL, Download, Open, Delete, Actions, Create folder, and Upload.

21. Visual Analysis using the AWS Quick Sight

- Give, Quick Sight access to AWS Services.

QuickSight access to AWS services

Make your existing AWS data and users available in QuickSight. [Learn more](#)

IAM Role

- Use QuickSight-managed role (default)
- Use an existing role

Allow access and autodiscovery for these resources

- Amazon Redshift
- Amazon RDS
- IAM
- Amazon S3 (7 buckets selected)
[Select S3 buckets](#)
- Amazon Athena
Make sure you've chosen the right Amazon S3 buckets for QuickSight access
- Amazon S3 Storage Analytics

- Choose dataset from the Aws Athena as data source.

QuickSight

Datasets

SPICE capacity for this region: Auto-purchase enabled

Create a Dataset

FROM NEW DATA SOURCES

The screenshot shows a grid of nine data source options:

- Upload a file (.csv, .tsv, .clif, .etl, .axlsx, .json)
- Salesforce (Connect to Salesforce)
- S3 Analytics
- S3
- Athena
- RDS
- Redshift (Auto-discovered)
- Redshift (Manual connect)
- MySQL

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

Choose your table

youtube_analytics_dashboard

Catalog: contain sets of databases.

AwsDataCatalog

Database: contain sets of tables.

db_analytics

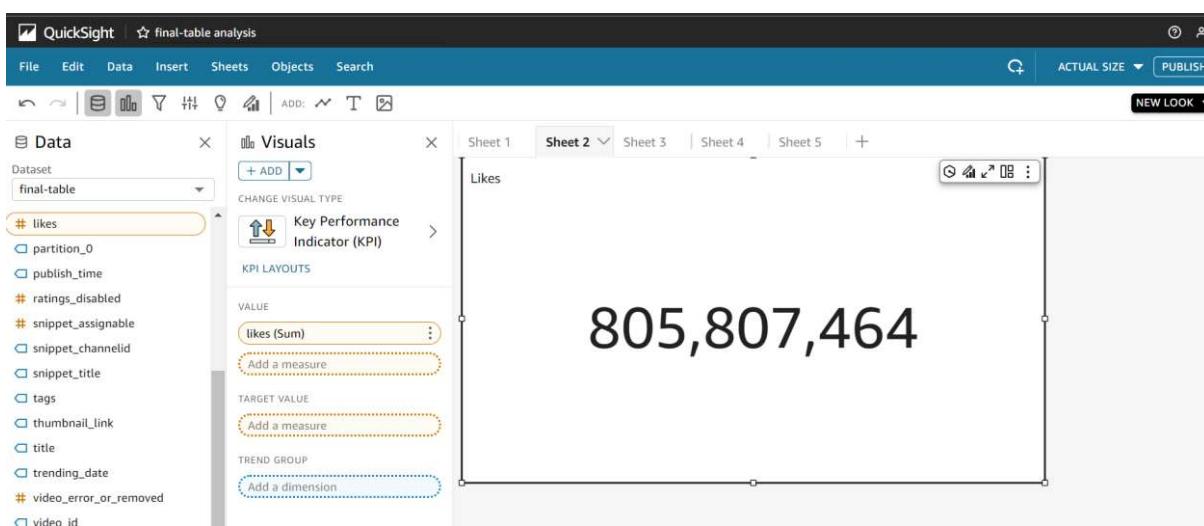
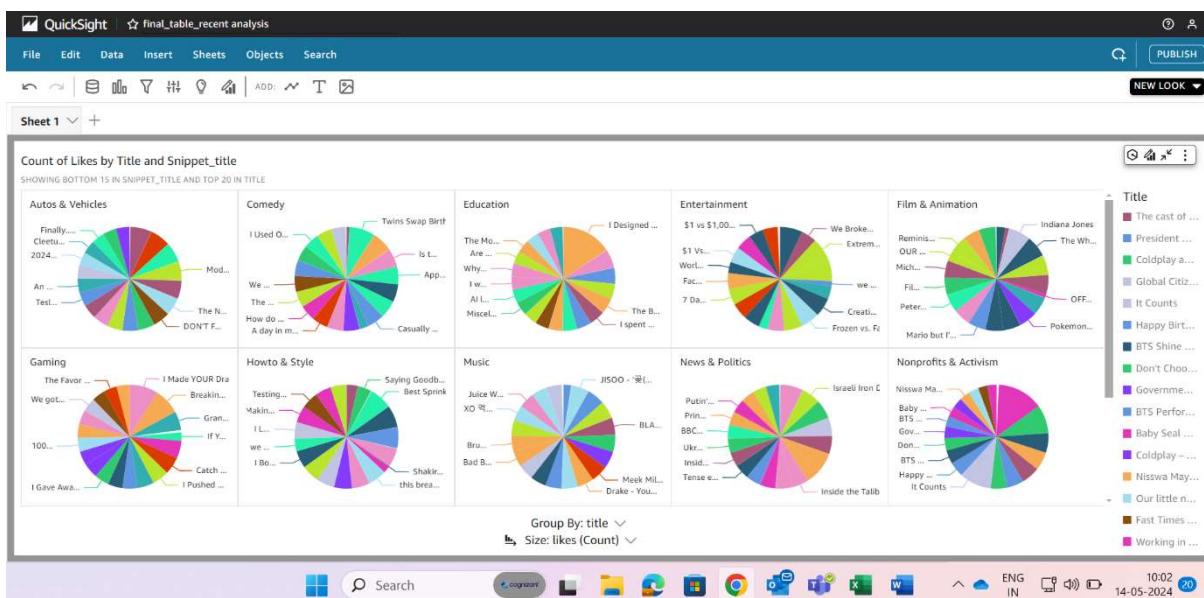
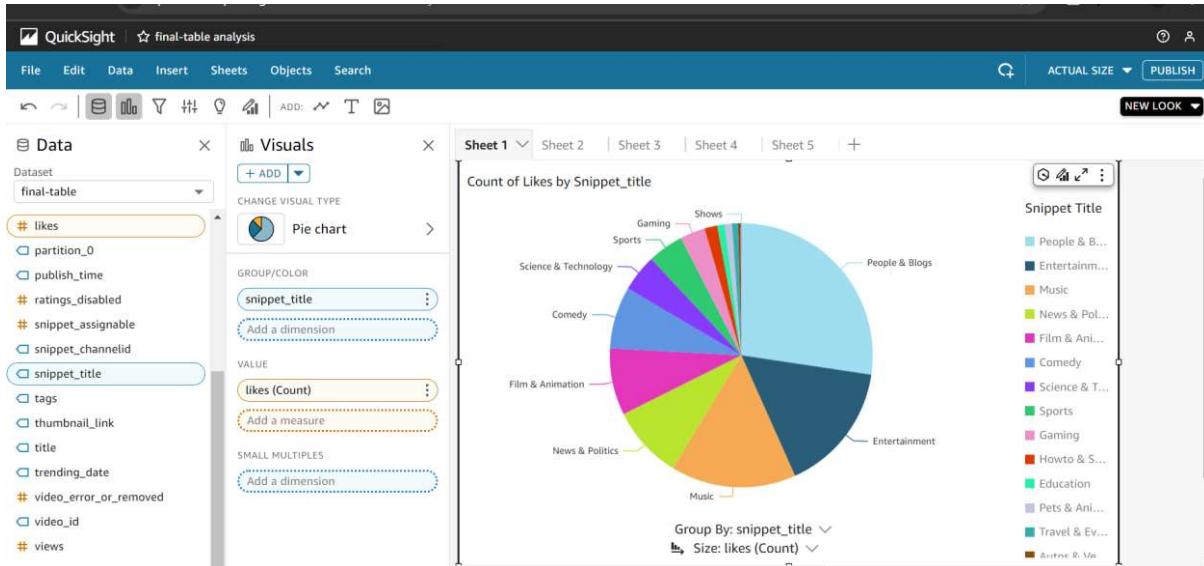
Tables: contain the data you can visualize.

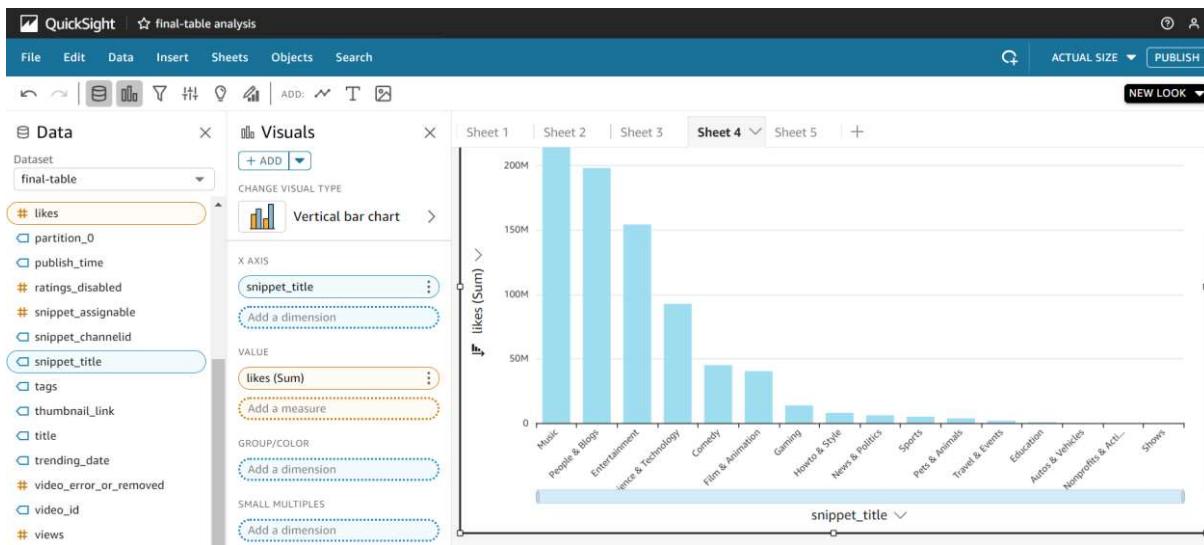
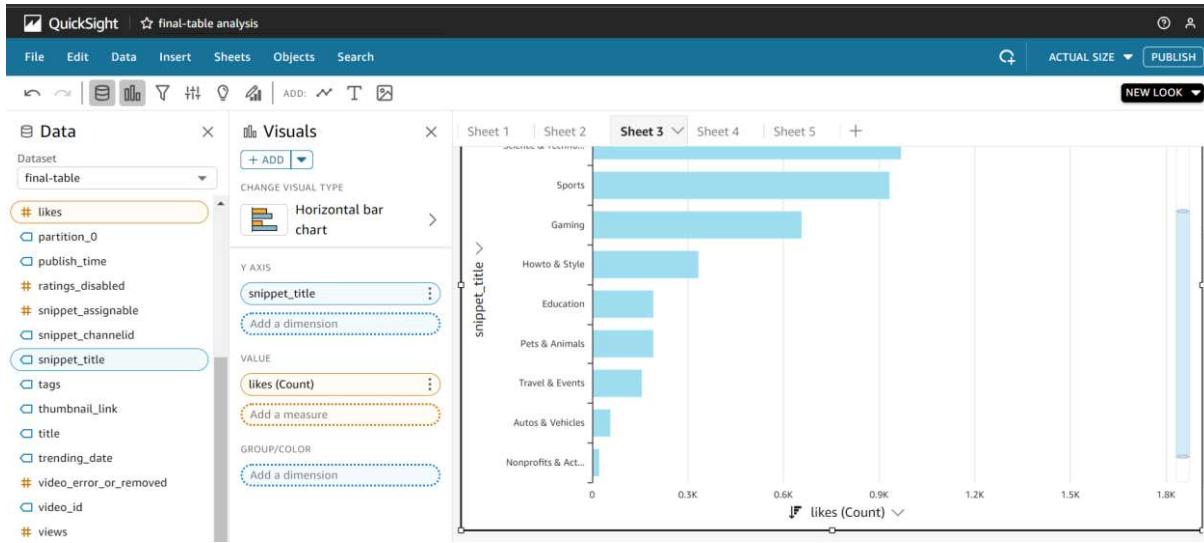
analytics_table

Edit/Preview data Use custom SQL Select

The dialog is titled "Choose your table" and shows the path selected: youtube_analytics_dashboard > AwsDataCatalog > db_analytics > analytics_table. At the bottom, there are three buttons: "Edit/Preview data", "Use custom SQL", and a large blue "Select" button.

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





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