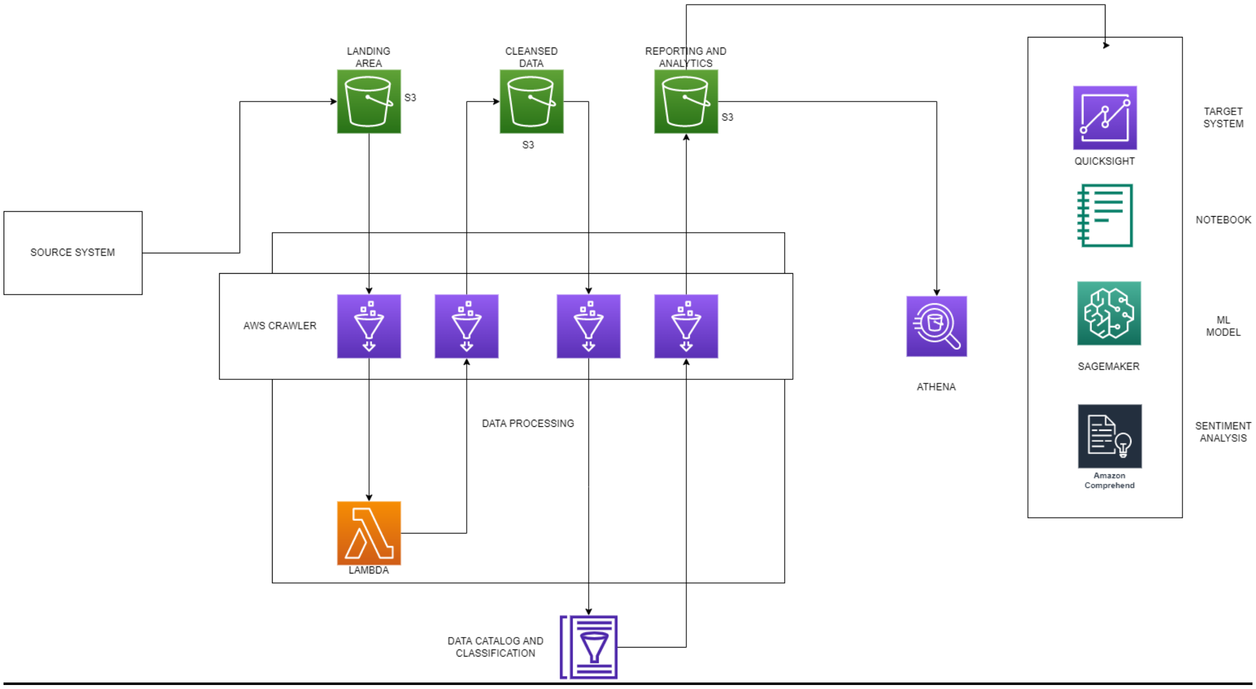
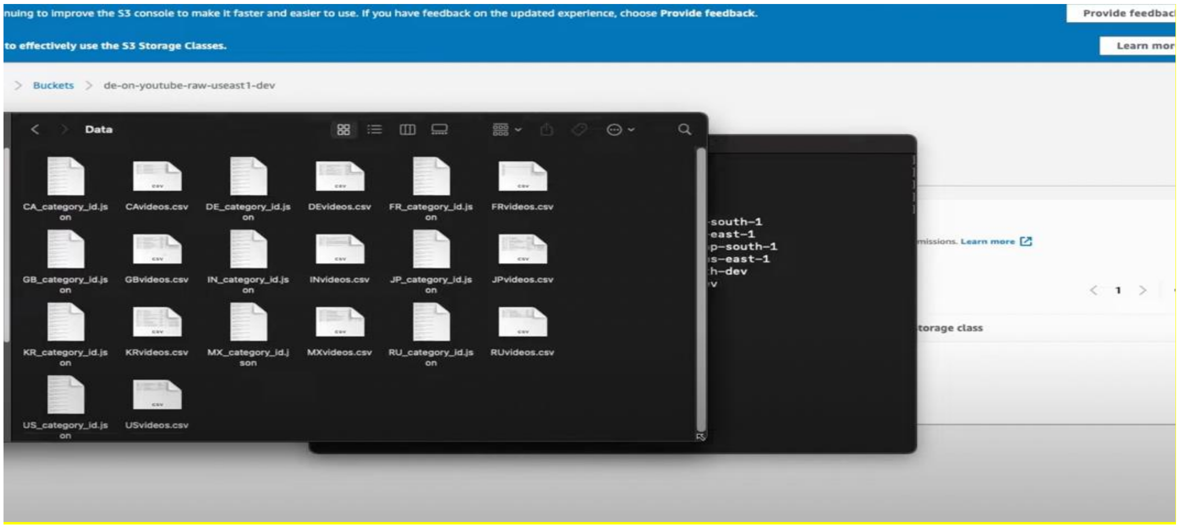
<u>Final Draft: Data Warehousing & Business Intelligence</u>

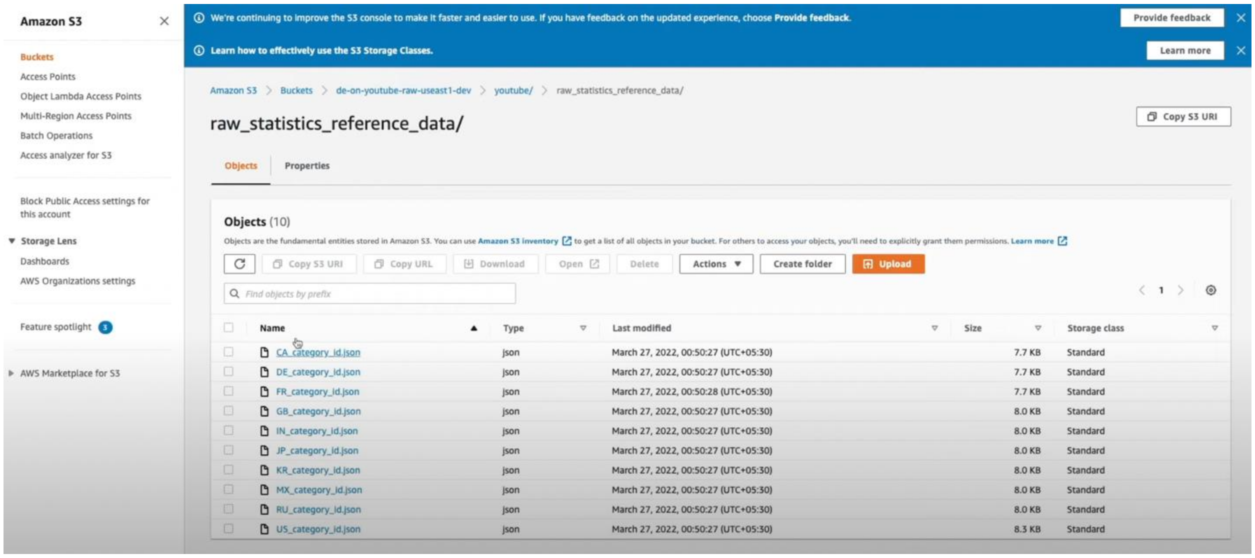
Project Group: 6

Ashok Thiruvengadam, Snehal Rajwar, Vidyun Akila Sundhara Raaman, Megha Batra



Step-1: Loading Data: We are loading the data from our API source data, which is in the form of JSON, CSV (Semi-structured Data). We are loading the source data into the AWS S3 bucket 1. This acts as the staging area. We are using the Extract, Load, and Transformation process.





We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose Provide feedback.

Learn how to effectively use the S3 Storage Classes.

Amazon S3 > Buckets > de-on-youtube-raw-useast1-dev > youtube/ > raw_statistics_reference_data/

raw_statistics_reference_data/

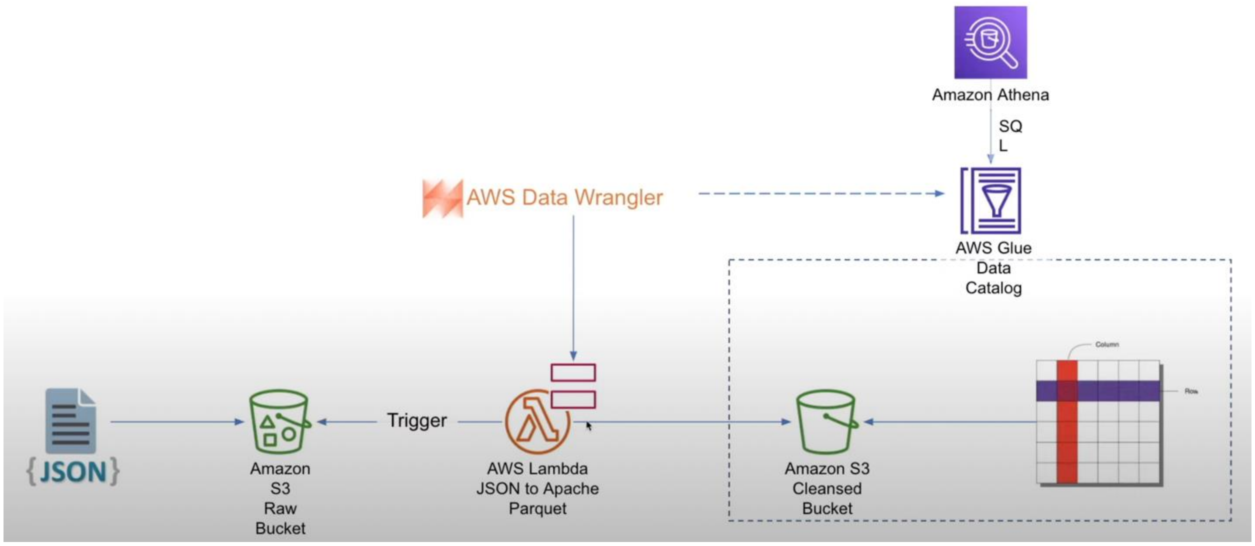
Objects (10)

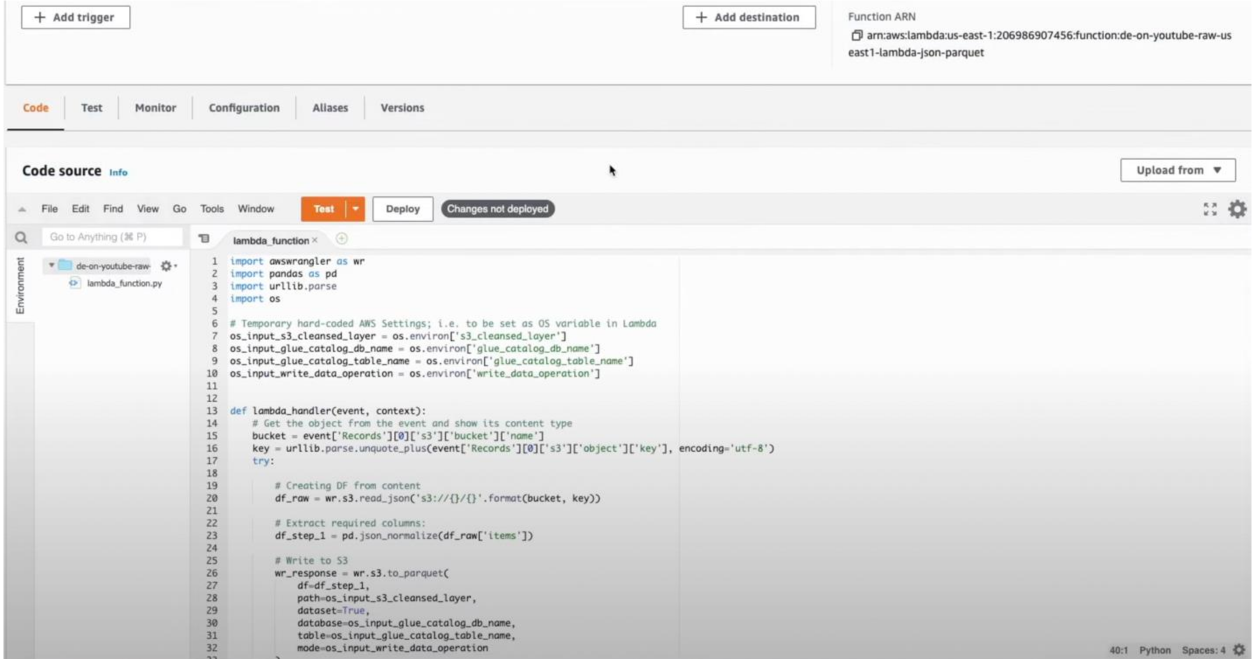
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
CA_category_id.json	json	March 27, 2022, 00:50:27 (UTC+05:30)	7.7 KB	Standard
DE_category_id.json	json	March 27, 2022, 00:50:27 (UTC+05:30)	7.7 KB	Standard
FR_category_id.json	json	March 27, 2022, 00:50:28 (UTC+05:30)	7.7 KB	Standard
GB_category_id.json	json	March 27, 2022, 00:50:27 (UTC+05:30)	8.0 KB	Standard
IN_category_id.json	json	March 27, 2022, 00:50:27 (UTC+05:30)	8.0 KB	Standard
JP_category_id.json	json	March 27, 2022, 00:50:27 (UTC+05:30)	8.0 KB	Standard
KR_category_id.json	json	March 27, 2022, 00:50:27 (UTC+05:30)	8.0 KB	Standard
MX_category_id.json	json	March 27, 2022, 00:50:27 (UTC+05:30)	8.0 KB	Standard
RU_category_id.json	json	March 27, 2022, 00:50:27 (UTC+05:30)	8.0 KB	Standard
US_category_id.json	json	March 27, 2022, 00:50:27 (UTC+05:30)	8.3 KB	Standard

Step 2: We are creating a data lake (centralized repository) that stores all the structured and unstructured data. We have JSON files that cannot be queried in AWS Athena. So, we are transforming the data from JSON to parquet file format using AWS Lambda. In AWS Lambda, we have used python to transform the data.

Data Cleansing
Semi-structured data to Structured pipeline

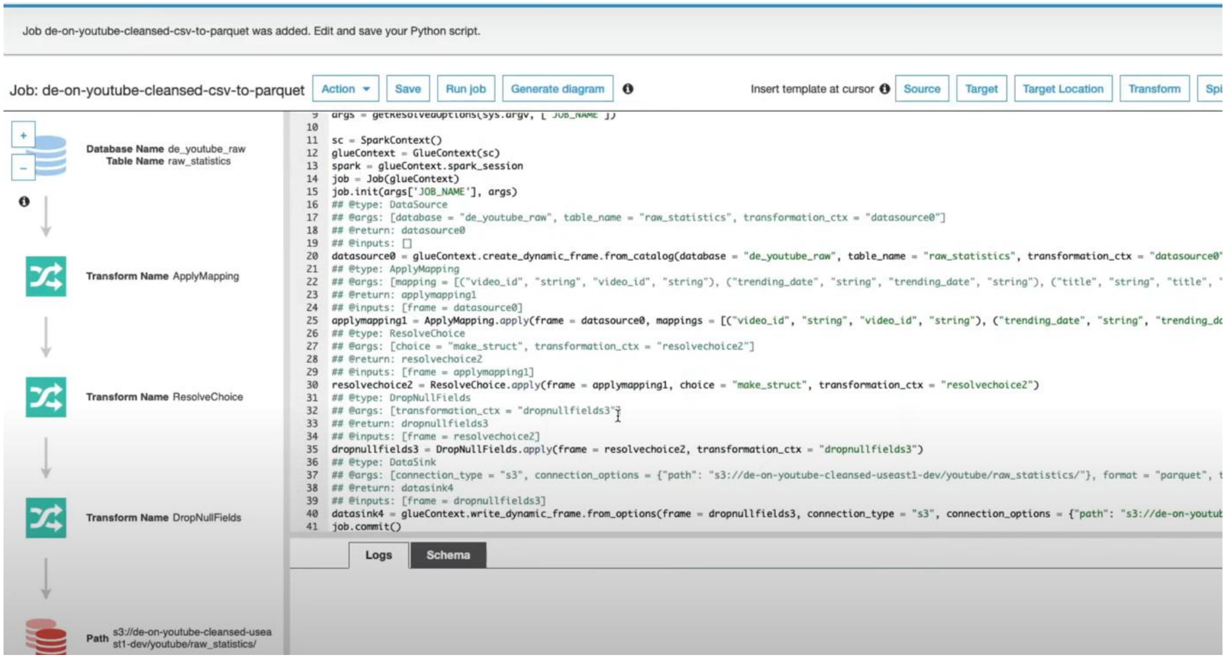




Function ARN
arn:aws:lambda:us-east-1:206986907456:function:de-on-youtube-raw-us east1-lambda-json-parquet

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

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


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

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

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

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

Step 3: Now, we have used the AWS glue crawler to move the transformed data to our clean S3 bucket which is the second bucket in our data lake. Then, we worked on the CSV files, then transformed the row data (CSV) and converted it into Apache Parquet format using AWS lambda and added it to the cleaned S3 second bucket. The second bucket includes the complete data which is cleaned and is in the Apache Parquet format.



Job de-on-youtube-cleansed-csv-to-parquet was added. Edit and save your Python script.

Job: de-on-youtube-cleansed-csv-to-parquet

Database Name de_youtube_raw
Table Name raw_statistics

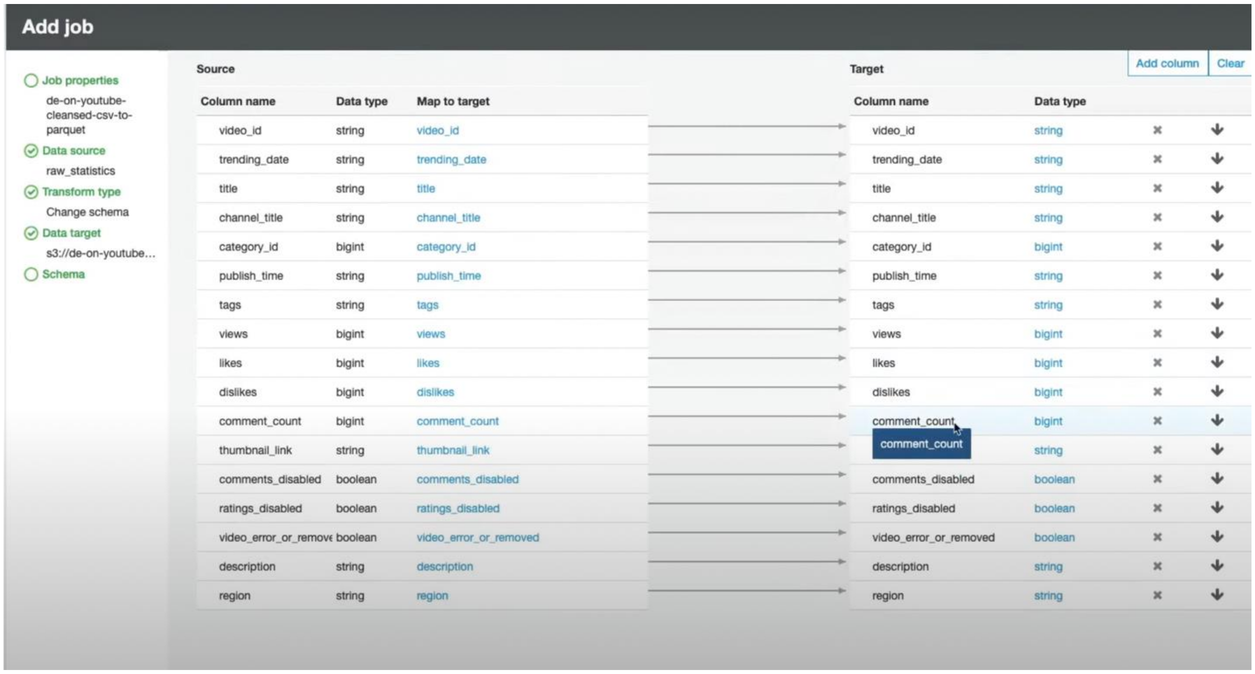
Transform Name ApplyMapping

Transform Name ResolveChoice

Transform Name DropNullFields

Path s3://de-on-youtube-cleansed-useast1-dev/youtube/raw_statistics/

```
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)
## @type: DataSource
## @args: [database = "de_youtube_raw", table_name = "raw_statistics", transformation_ctx = "datasource0"]
## @return: datasource0
## @inputs: []
datasource0 = glueContext.create_dynamic_frame.from_catalog(database = "de_youtube_raw", table_name = "raw_statistics", transformation_ctx = "datasource0"
## @type: ApplyMapping
## @args: [mapping = [("video_id", "string", "video_id", "string"), ("trending_date", "string", "trending_date", "string"), ("title", "string", "title", "
## @return: applymapping1
## @inputs: [frame = datasource0]
applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [("video_id", "string", "video_id", "string"), ("trending_date", "string", "trending_da
## @type: ResolveChoice
## @args: [choice = "make_struct", transformation_ctx = "resolvechoice2"]
## @return: resolvechoice2
## @inputs: [frame = applymapping1]
resolvechoice2 = ResolveChoice.apply(frame = applymapping1, choice = "make_struct", transformation_ctx = "resolvechoice2")
## @type: DropNullFields
## @args: [transformation_ctx = "dropnullfields3"]
## @return: dropnullfields3
## @inputs: [frame = resolvechoice2]
dropnullfields3 = DropNullFields.apply(frame = resolvechoice2, transformation_ctx = "dropnullfields3")
## @type: DataSink
## @args: [connection_type = "s3", connection_options = {"path": "s3://de-on-youtube-cleansed-useast1-dev/youtube/raw_statistics/"}, format = "parquet", t
## @return: datasink4
## @inputs: [frame = dropnullfields3]
datasink4 = glueContext.write_dynamic_frame.from_options(frame = dropnullfields3, connection_type = "s3", connection_options = {"path": "s3://de-on-youtub
job.commit()
```



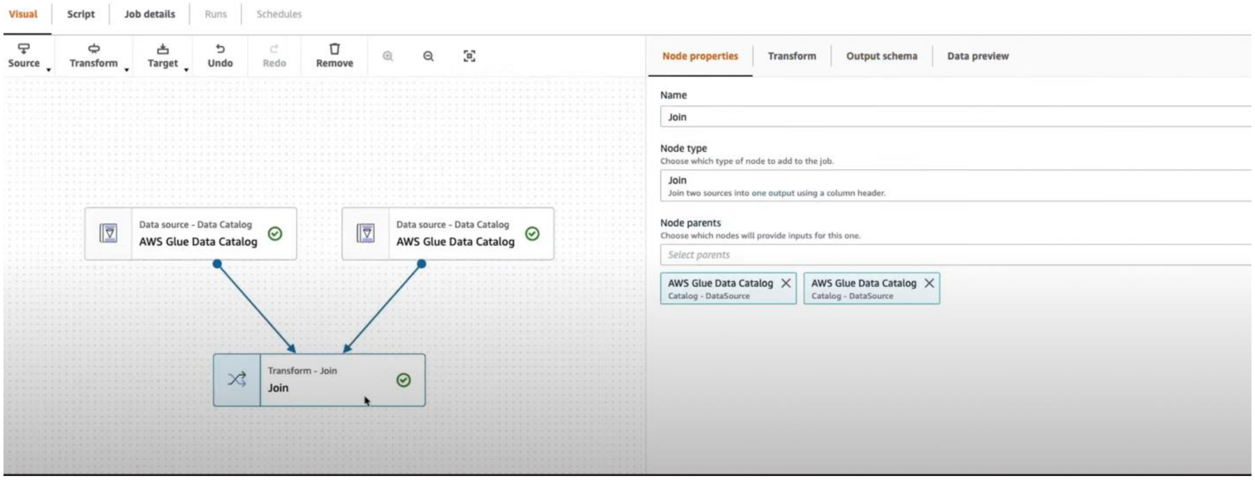
Add job

○ Job properties
de-on-youtube-cleansed-csv-to-parquet
✓ Data source
raw_statistics
✓ Transform type
Change schema
✓ Data target
s3://de-on-youtube...
○ Schema

Source: Column name	Data type	Map to target	Target: Column name	Data type
video_id	string	video_id	video_id	string
trending_date	string	trending_date	trending_date	string
title	string	title	title	string
channel_title	string	channel_title	channel_title	string
category_id	bigint	category_id	category_id	bigint
publish_time	string	publish_time	publish_time	string
tags	string	tags	tags	string
views	bigint	views	views	bigint
likes	bigint	likes	likes	bigint
dislikes	bigint	dislikes	dislikes	bigint
comment_count	bigint	comment_count	comment_count	bigint
thumbnail_link	string	thumbnail_link		string
comments_disabled	boolean	comments_disabled	comments_disabled	boolean
ratings_disabled	boolean	ratings_disabled	ratings_disabled	boolean
video_error_or_remove	boolean	video_error_or_removed	video_error_or_removed	boolean
description	string	description	description	string
region	string	region	region	string

Step 4: Automation to load future incoming data using triggers (we are working on it)

Step 5: From the cleaned second S3 bucket we have used the AWS Glue studio catalog to create a data catalog where we joined the required tables for further querying and then added it to reporting bucket to perform reporting and BI.



Name

Join

Node type

Choose which type of node to add to the job.

Join

Join two sources into one output using a column header.

Node parents

Choose which nodes will provide inputs for this one.

Select parents

AWS Glue Data Catalog ✕
Catalog - DataSource

AWS Glue Data Catalog ✕
Catalog - DataSource

Step 6: We are planning to use AWS SageMaker to deploy ML model for categorization of YouTube videos based on description and AWS Comprehend to implement sentiment analysis (we are working on it)

Step 7: We are planning to use AWS QuickSight to do the business intelligence part (we are working on it)

1. We will make use of a word cloud, bar charts, line charts, and bubble charts to Count number of likes, Count number of likes by snippet title, Sum of views by snippet title, Sum of views by region
2. We will check which category of videos which had the most number of likes, the most number of views, which category of videos was popular in which region and categorizing based on comments and statistics.