**Data Engineering YouTube Analysis Project by Prakash**

**Overview**

This project aims to securely manage, streamline, and perform analysis on the structured and semi-structured YouTube videos data based on the video categories and the trending metrics.

**Project Goals**

1. Data Ingestion — Build a mechanism to ingest data from different sources
2. ETL System — We are getting data in raw format, transforming this data into the proper format.
3. Data lake — We will be getting data from multiple sources so we need centralized repo to store them
4. Scalability — As the size of our data increases, we need to make sure our system scales with it
5. Cloud — We can’t process vast amounts of data on our local computer so we need to use the cloud, in this case, we will use AWS
6. Reporting — Build a dashboard to get answers to the question we asked earlier

**Services we will be using**

1. Amazon S3: Amazon S3 is an object storage service that provides manufacturing scalability, data availability, security, and performance.
2. AWS IAM: This is nothing but identity and access management which enables us to manage access to AWS services and resources securely.
3. QuickSight: Amazon QuickSight is a scalable, serverless, embeddable, machine learning-powered business intelligence (BI) service built for the cloud.
4. AWS Glue: A serverless data integration service that makes it easy to discover, prepare, and combine data for analytics, machine learning, and application development.
5. AWS Lambda: Lambda is a computing service that allows programmers to run code without creating or managing servers.
6. AWS Athena: Athena is an interactive query service for S3 in which there is no need to load data it stays in S3.

**Dataset Used**

This Kaggle dataset contains statistics (CSV files) on daily popular YouTube videos over the course of many months. There are up to 200 trending videos published every day for many locations. The data for each region is in its own file. The video title, channel title, publication time, tags, views, likes and dislikes, description, and comment count are among the items included in the data. A category\_id field, which differs by area, is also included in the JSON file linked to the region.

Data set download link

[Trending YouTube Video Statistics (kaggle.com)](https://www.kaggle.com/datasets/datasnaek/youtube-new)

**Data Architecture Diagram**

A screenshot of a software application

Description automatically generated

**lambda\_function.py**

import awswrangler as wr

import pandas as pd

import urllib.parse

import os

os\_input\_s3\_cleansed\_layer = os.environ['s3\_cleansed\_layer']

os\_input\_glue\_catalog\_db\_name = os.environ['glue\_catalog\_db\_name']

os\_input\_glue\_catalog\_table\_name = os.environ['glue\_catalog\_table\_name']

os\_input\_write\_data\_operation = os.environ['write\_data\_operation']

def lambda\_handler(event, context):

# Get the object from the event and show its content type

bucket = event['Records'][0]['s3']['bucket']['name']

key = urllib.parse.unquote\_plus(event['Records'][0]['s3']['object']['key'], encoding='utf-8')

try:

# Creating DF from content

df\_raw = wr.s3.read\_json('s3://{}/{}'.format(bucket, key))

# Extract required columns:

df\_step\_1 = pd.json\_normalize(df\_raw['items'])

# Write to S3

wr\_response = wr.s3.to\_parquet(

df=df\_step\_1,

path=os\_input\_s3\_cleansed\_layer,

dataset=True,

database=os\_input\_glue\_catalog\_db\_name,

table=os\_input\_glue\_catalog\_table\_name,

mode=os\_input\_write\_data\_operation

)

return wr\_response

except Exception as e:

print(e)

print('Error getting object {} from bucket {}. Make sure they exist and your bucket is in the same region as this function.'.format(key, bucket))

raise e

**pyspark\_code.py**

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

from awsglue.dynamicframe import DynamicFrame

## @params: [JOB\_NAME]

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 = "db\_youtube\_raw", table\_name = "raw\_statistics", transformation\_ctx = "datasource0"]

## @return: datasource0

## @inputs: []

predicate\_pushdown = "region in ('ca','gb','us')"

datasource0 = glueContext.create\_dynamic\_frame.from\_catalog(database = "db\_youtube\_raw", table\_name = "raw\_statistics", transformation\_ctx = "datasource0", push\_down\_predicate = predicate\_pushdown)

## @type: ApplyMapping

## @args: [mapping = [("video\_id", "string", "video\_id", "string"), ("trending\_date", "string", "trending\_date", "string"), ("title", "string", "title", "string"), ("channel\_title", "string", "channel\_title", "string"), ("category\_id", "long", "category\_id", "long"), ("publish\_time", "string", "publish\_time", "string"), ("tags", "string", "tags", "string"), ("views", "long", "views", "long"), ("likes", "long", "likes", "long"), ("dislikes", "long", "dislikes", "long"), ("comment\_count", "long", "comment\_count", "long"), ("thumbnail\_link", "string", "thumbnail\_link", "string"), ("comments\_disabled", "boolean", "comments\_disabled", "boolean"), ("ratings\_disabled", "boolean", "ratings\_disabled", "boolean"), ("video\_error\_or\_removed", "boolean", "video\_error\_or\_removed", "boolean"), ("description", "string", "description", "string"), ("region", "string", "region", "string")], transformation\_ctx = "applymapping1"]

## @return: applymapping1

## @inputs: [frame = datasource0]

applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [("video\_id", "string", "video\_id", "string"), ("trending\_date", "string", "trending\_date", "string"), ("title", "string", "title", "string"), ("channel\_title", "string", "channel\_title", "string"), ("category\_id", "long", "category\_id", "long"), ("publish\_time", "string", "publish\_time", "string"), ("tags", "string", "tags", "string"), ("views", "long", "views", "long"), ("likes", "long", "likes", "long"), ("dislikes", "long", "dislikes", "long"), ("comment\_count", "long", "comment\_count", "long"), ("thumbnail\_link", "string", "thumbnail\_link", "string"), ("comments\_disabled", "boolean", "comments\_disabled", "boolean"), ("ratings\_disabled", "boolean", "ratings\_disabled", "boolean"), ("video\_error\_or\_removed", "boolean", "video\_error\_or\_removed", "boolean"), ("description", "string", "description", "string"), ("region", "string", "region", "string")], transformation\_ctx = "applymapping1")

## @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://bigdata-on-youtube-cleansed-euwest1-14317621-dev/youtube/raw\_statistics/"}, format = "parquet", transformation\_ctx = "datasink4"]

## @return: datasink4

## @inputs: [frame = dropnullfields3]

#MAKE SURE YOU COPY ONLY WHAT IS NEEDED

datasink1 = dropnullfields3.toDF().coalesce(1)

df\_final\_output = DynamicFrame.fromDF(datasink1, glueContext, "df\_final\_output")

datasink4 = glueContext.write\_dynamic\_frame.from\_options(frame = df\_final\_output, connection\_type = "s3", connection\_options = {"path": "s3://de-on-youtube-cleansed-useast1-dev/youtube/raw\_statistics/", "partitionKeys": ["region"]}, format = "parquet", transformation\_ctx = "datasink4")

job.commit()

**s3\_cli\_command.sh**

**#Replace It With Your Bucket Name**

**# To copy all JSON Reference data to same location:**

**aws s3 cp . s3://de-on-youtube-raw-useast1-dev/youtube/raw\_statistics\_reference\_data/ --recursive --exclude "\*" --include "\*.json"**

**# To copy all data files to its own location, following Hive-style patterns:**

**aws s3 cp CAvideos.csv s3://de-on-youtube-raw-useast1-dev/youtube/raw\_statistics/region=ca/**

**aws s3 cp DEvideos.csv s3://de-on-youtube-raw-useast1-dev/youtube/raw\_statistics/region=de/**

**aws s3 cp FRvideos.csv s3://de-on-youtube-raw-useast1-dev/youtube/raw\_statistics/region=fr/**

**aws s3 cp GBvideos.csv s3://de-on-youtube-raw-useast1-dev/youtube/raw\_statistics/region=gb/**

**aws s3 cp INvideos.csv s3://de-on-youtube-raw-useast1-dev/youtube/raw\_statistics/region=in/**

**aws s3 cp JPvideos.csv s3://de-on-youtube-raw-useast1-dev/youtube/raw\_statistics/region=jp/**

**aws s3 cp KRvideos.csv s3://de-on-youtube-raw-useast1-dev/youtube/raw\_statistics/region=kr/**

**aws s3 cp MXvideos.csv s3://de-on-youtube-raw-useast1-dev/youtube/raw\_statistics/region=mx/**

**aws s3 cp RUvideos.csv s3://de-on-youtube-raw-useast1-dev/youtube/raw\_statistics/region=ru/**

**aws s3 cp USvideos.csv s3://de-on-youtube-raw-useast1-dev/youtube/raw\_statistics/region=us/**