Project Title: YouTube Content Insights: Data Engineering and Analysis

Project– Second Update

Description:

The Data Engineering YouTube Analysis Project aims to securely manage, streamline, and perform analysis on structured and semi-structured YouTube video data. We have successfully initiated the project with the setup of our cloud infrastructure using AWS (Amazon Web Services) and have begun the process of data ingestion with a focus on video categories and trending metrics.

Recent progress:

Building a data pipeline using AWS Glue, Athena, Lambda, and SQL involves several steps.

**AWS Infrastructure Setup:**

We have collaborated on setting up the AWS account and installing the AWS CLI for programmatic interactions with AWS services.

**S3 Bucket Creation and Data Upload:**

An S3 bucket has been created in the AWS account for storage, and we have uploaded the "Trending YouTube Video Statistics" dataset. This dataset was available on Kaggle. This dataset includes daily records of trending YouTube videos from multiple regions (US, GB, DE, CA, FR, RU, MX, KR, JP, and IN), providing a diverse and rich source for our analysis.

A screenshot of a computer

Description automatically generated

**Dataset Details:**

* The dataset encompasses a wide array of data points, including video title, channel title, publish time, tags, views, likes and dislikes, description, and comment count.
* It also features a category\_id field for each region, linking to JSON files that provide deeper insights into video categorization.

Recent progress:

**AWS Glue Catalog Building:**

Our initial step in developing the ETL system was the creation of an AWS Glue Catalog. This foundational task was essential for the discovery and management of our data's metadata. Establishing the Glue Catalog has laid the groundwork for our subsequent ETL operations, enabling a streamlined approach to data transformation and analysis.

A screenshot of a computer

Description automatically generated

**Crawler Configuration:**

By configuring a crawler to process JSON files stored in our S3 bucket, we initiated the "Extract" phase of our ETL process. This crucial step allows us to understand the structure and schema of our data, which is vital for planning our transformation and loading strategies. When creating the crawler, we have defined the data source path as:

s3://ashade-on-youtube-raw-useast2-dev/youtube/raw\_statistics\_reference\_data/

The above is the S3 bucket path in which the raw data is stored in json format.

A screenshot of a computer

Description automatically generated

**Data Transformation: (Using Athena to Query Data)**

A diagram of a data pipeline

Description automatically generated

We encountered challenges with data querying, particularly with Athena's processing of our JSON data's initial structure. This challenge led us to transform our JSON data into the Apache Parquet format, marking a significant step in the "Transform" phase.

A screenshot of a computer error message

Description automatically generated

The reason behind this is the structure of the data, which AWS crawler is not able to understand. Some preprocessing is needed here in order to query the data properly.

We have solved the above error using the below Lambda function for this conversion to ensure data is in an optimal format for analysis.

import awswrangler as wr

import pandas as pd

import urllib.parse

import os

import logging

# Configure logging

logger = logging.getLogger()

logger.setLevel(logging.INFO)

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')

s3\_path = f's3://{bucket}/{key}'

try:

# Log the start of the process

logger.info(f"Starting processing for {s3\_path}")

# Creating DF from content

df\_raw = wr.s3.read\_json(s3\_path)

# Log successful read

logger.info(f"Successfully read JSON from {s3\_path}")

# Extract required columns:

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

# Log successful transformation

logger.info("Successfully transformed JSON to DataFrame")

# 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

)

# Log successful write

logger.info(f"Successfully wrote Parquet to {os\_input\_s3\_cleansed\_layer}. Response: {wr\_response}")

return wr\_response

except Exception as e:

logger.error("Error during processing", exc\_info=True)

raise e

**List of environment variables**

* glue\_catalog\_db\_name = db\_youtube\_cleaned
* glue\_catalog\_table\_name = cleaned\_statistics\_reference\_data
* s3\_cleansed\_layer = s3://ashade-on-youtube-cleansed-raw-useast2-dev/youtube
* write\_data\_operation = append

The above Lambda function reads the raw data from the S3 locations and transforms the JSON files into Parquet format and writes the cleansed files to the another S3bucket.

A screenshot of a computer

Description automatically generated

Figure1: Retrieval of rows from the cleansed data(Youtube)

A screenshot of a computer

Description automatically generated

Figure 2: Retrieval of rows from th cleansed data(Youtube) -ca\_category\_id\_json