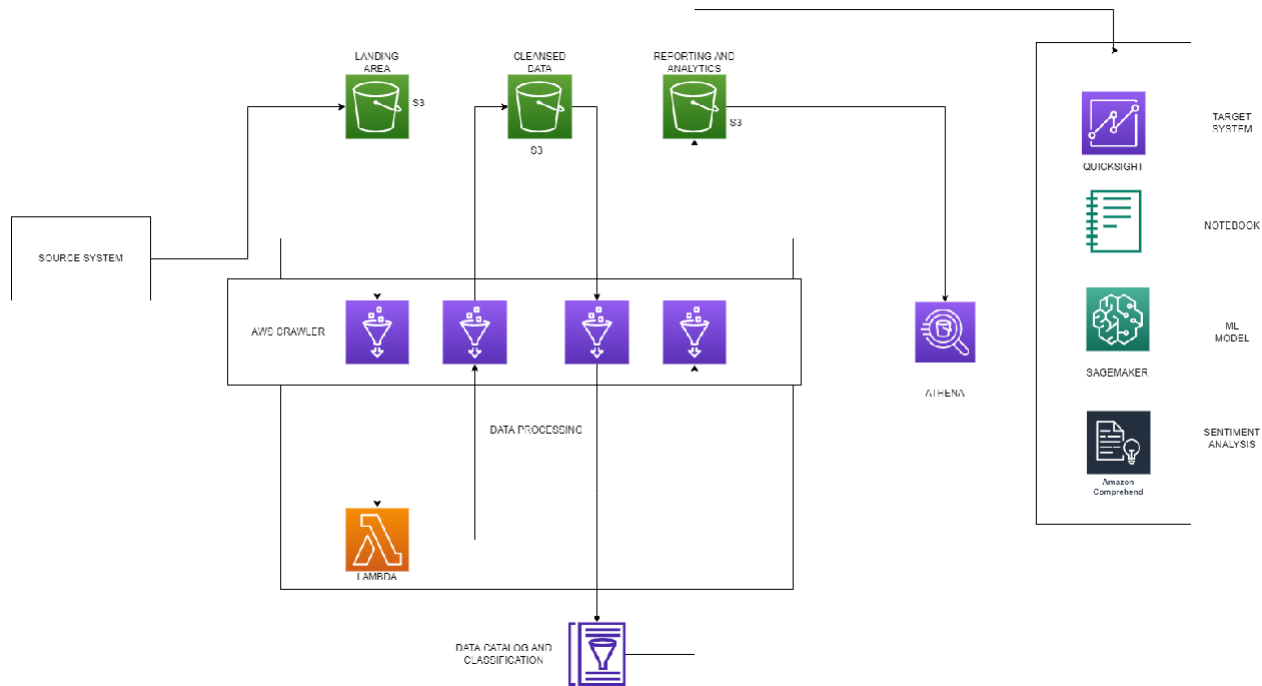


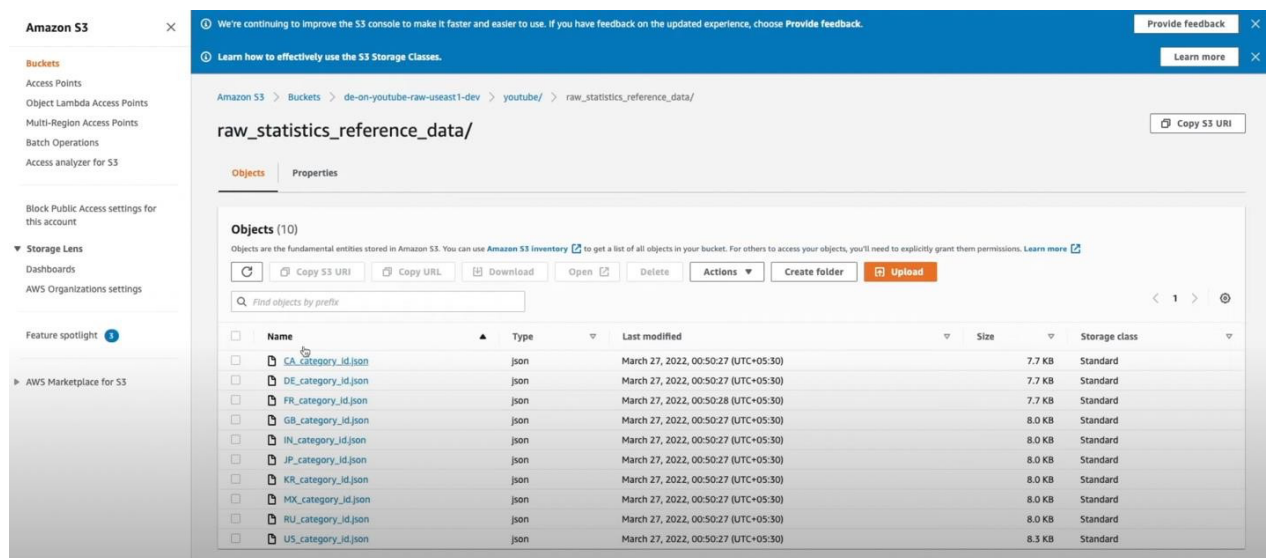
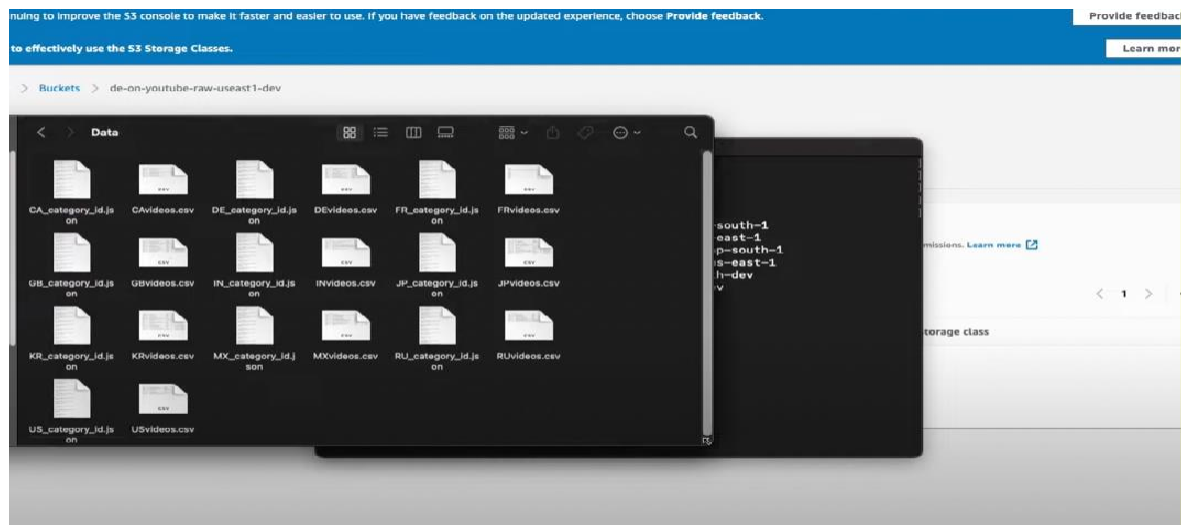
Final Draft: Data Warehousing & Business Intelligence

Project Group: 6

Nithya Rani Vuddagiri, Vikas Sunkari

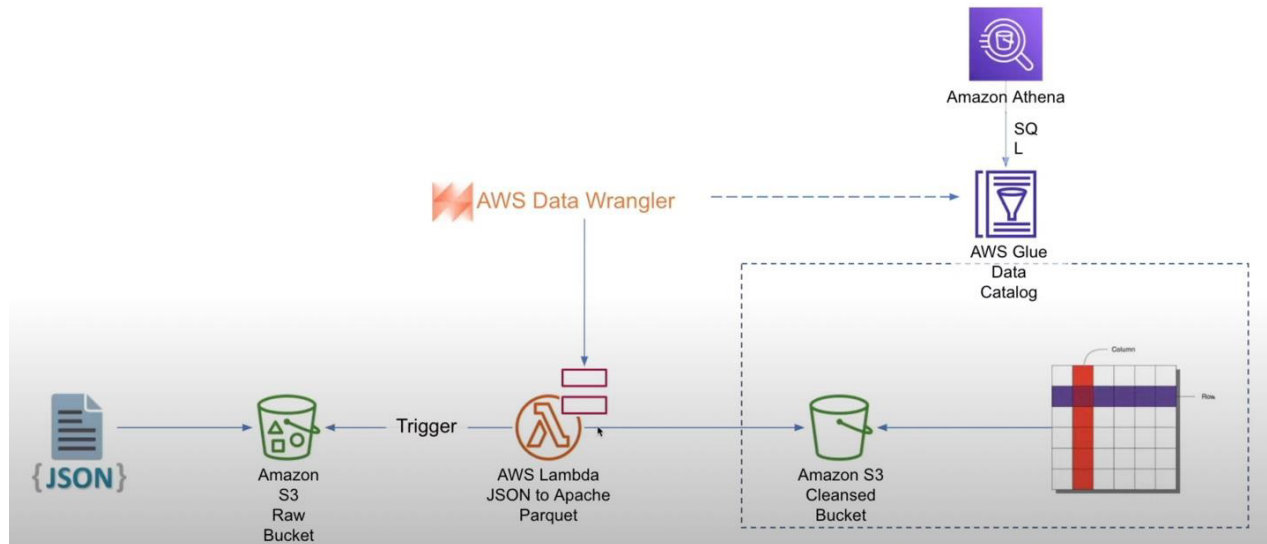


Step-1: Loading Data: We are loading the data from our API source data, which is in the form of JSON, CSV (Semi-structured Data). We are loading the source data into the AWS S3 bucket 1. This acts as the staging area. We are using the Extract, Load, and Transformation process.



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



+ Add trigger

+ Add destination

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

Code

Test

Monitor

Configuration

Aliases

Versions

Code source

Info

Upload from

File

Edit

Find

View

Go

Tools

Window

Test

Deploy

Changes not deployed

Go to Anything (⌘ P)

Environment

de-on-youtube-raw

lambda_function.py

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

40:1 Python Spaces:4

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 Action Save Run job Generate diagram Insert template at cursor Source Target Target Location Transform Split

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/

```

9  args = get_resolve_options(sys.argv, ['JOB_NAME'])
10
11  sc = SparkContext()
12  glueContext = GlueContext(sc)
13  spark = glueContext.spark_session
14  job = Job(glueContext)
15  job.init(args['JOB_NAME'], args)
16  ## @type: DataSource
17  ## @args: [database = "de_youtube_raw", table_name = "raw_statistics", transformation_ctx = "datasource0"]
18  ## @return: DataSource
19  ## @type: DataSource
20  datasource0 = glueContext.create_dynamic_frame.from_catalog(database = "de_youtube_raw", table_name = "raw_statistics", transformation_ctx = "datasource0")
21  ## @type: ApplyMapping
22  ## @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", "string", "category_id", "string"}, {"publish_time", "string", "publish_time", "string"}, {"tags", "string", "tags", "string"}, {"views", "string", "views", "string"}, {"likes", "string", "likes", "string"}, {"dislikes", "string", "dislikes", "string"}, {"comment_count", "string", "comment_count", "string"}, {"thumbnail_link", "string", "thumbnail_link", "string"}, {"comments_disabled", "string", "comments_disabled", "string"}, {"ratings_disabled", "string", "ratings_disabled", "string"}, {"video_error_or_removed", "string", "video_error_or_removed", "string"}, {"description", "string", "description", "string"}, {"region", "string", "region", "string"}]]
23  ## @return: ApplyMapping
24  ## @inputs: [frame = datasource0]
25  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", "string", "category_id", "string"}, {"publish_time", "string", "publish_time", "string"}, {"tags", "string", "tags", "string"}, {"views", "string", "views", "string"}, {"likes", "string", "likes", "string"}, {"dislikes", "string", "dislikes", "string"}, {"comment_count", "string", "comment_count", "string"}, {"thumbnail_link", "string", "thumbnail_link", "string"}, {"comments_disabled", "string", "comments_disabled", "string"}, {"ratings_disabled", "string", "ratings_disabled", "string"}, {"video_error_or_removed", "string", "video_error_or_removed", "string"}, {"description", "string", "description", "string"}, {"region", "string", "region", "string"}])
26  ## @type: ResolveChoice
27  ## @args: [choice = "make_struct", transformation_ctx = "resolvechoice2"]
28  ## @return: ResolveChoice
29  ## @inputs: [frame = applymapping1]
30  resolvechoice2 = ResolveChoice.apply(frame = applymapping1, choice = "make_struct", transformation_ctx = "resolvechoice2")
31  ## @type: DropNullFields
32  ## @args: [transformation_ctx = "dropnullfields3"]
33  ## @return: DropNullFields
34  ## @inputs: [frame = resolvechoice2]
35  dropnullfields3 = DropNullFields.apply(frame = resolvechoice2, transformation_ctx = "dropnullfields3")
36  ## @type: DataSink
37  ## @args: [connection_type = "s3", connection_options = {"path": "s3://de-on-youtube-cleansed-useast1-dev/youtube/raw_statistics/"}, format = "parquet", transformation_ctx = "datasink4"]
38  ## @return: DataSink
39  ## @inputs: [frame = dropnullfields3]
40  datasink4 = glueContext.write_dynamic_frame.from_options(frame = dropnullfields3, connection_type = "s3", connection_options = {"path": "s3://de-on-youtube-cleansed-useast1-dev/youtube/raw_statistics/"}, format = "parquet", transformation_ctx = "datasink4")
41  job.commit()

```

Logs Schema

Add job

☐ Job properties

☒ Data source

☒ Transform type

☒ Data target

☐ Schema

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

raw_statistics

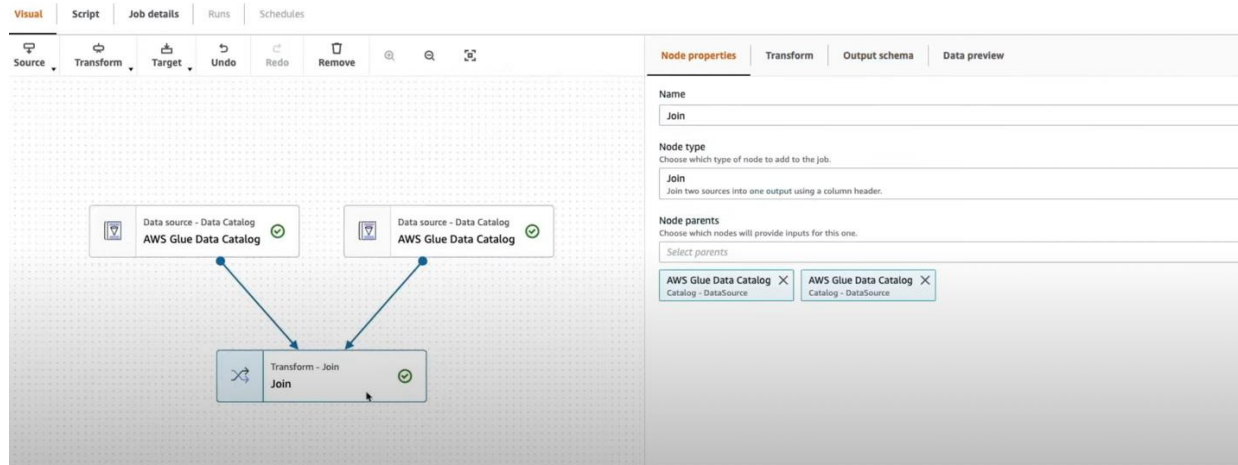
Change schema

s3://de-on-youtube...

Source			Target		
Column name	Data type	Map to 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	comment_count	string	×
comments_disabled	boolean	comments_disabled	comments_disabled	boolean	×
ratings_disabled	boolean	ratings_disabled	ratings_disabled	boolean	×
video_error_or_removed	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.



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.