YouTube-Analysis-Project-On-AWS

Overview:

The objective of this project is to handle structured and semi-structured YouTube video data in a safe manner, optimize it, and do analysis on it using trending metrics and video categories.

Dataset: https://www.kaggle.com/datasets/datasnaek/youtube-new

Goal:

1. **Data Ingestion**: Construct a system to take in information from many sources.
2. **ETL System**: We receive data in raw form and convert it to the appropriate format.
3. **Data lake**: To store the data that we will be obtaining from various sources, we need a centralized repository.
4. **Scalability**: We must ensure that our system can grow as our data grows.
5. **Cloud**: Since our local computer cannot handle processing large volumes of data, we must use the cloud; in this instance, AWS
6. **Reporting**: Create a dashboard to obtain the responses to the previous query.

AWS Services I will Use in this Project:

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.

Procedure:

1. Create IAM user

Username: de-YouTube-project
Permission policy: Administrator Access
generate access key and secret access key. The description for the access key is 'de-YouTube-project'.
To sign into an AWS account as an IAM user, you must have an account alias or an account ID for the AWS account. Account id is in the support center.

2. Setup AWS CLI

```
$aws configure
enter the access key and secret access key
#list all the s3 bucket
$aws s3 ls
```

3. Upload Data into S3

Create a new s3 buckets named 'youtube-de-raw-useast1-dev' and upload the data set into two folders one contains json format and another one contains csv format and following Hive-style patterns.

```
$cd Downloads/data
$ls
# To copy all JSON Reference data to same location:
$aws s3 cp . s3://youtube-de-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://youtube-de-raw-useast1-devri/youtube/raw_statistics/region=ca/
aws s3 cp DEvideos.csv s3://youtube-de-raw-useast1-devri/youtube/raw_statistics/region=de/
aws s3 cp FRvideos.csv s3://youtube-de-raw-useast1-devri/youtube/raw_statistics/region=fr/
aws s3 cp GBvideos.csv s3://youtube-de-raw-useast1-devri/youtube/raw_statistics/region=gb/
aws s3 cp INvideos.csv s3://youtube-de-raw-useast1-devri/youtube/raw_statistics/region=in/
aws s3 cp JPvideos.csv s3://youtube-de-raw-useast1-devri/youtube/raw_statistics/region=jp/
aws s3 cp KRvideos.csv s3://youtube-de-raw-useast1-devri/youtube/raw_statistics/region=kr/
aws s3 cp MXvideos.csv s3://youtube-de-raw-useast1-devri/youtube/raw_statistics/region=mx/
aws s3 cp RUvideos.csv s3://youtube-de-raw-useast1-devri/youtube/raw_statistics/region=ru/
aws s3 cp USvideos.csv s3://youtube-de-raw-useast1-devri/youtube/raw_statistics/region=us/
```

4. Build Glue Crawler and Catalog

Go to Glue Crawlers and create a crawler named 'youtube-de-raw-glue-catalog-1', then add a data source in our s3 bucket "**s3/youtube-de-raw-useast1-devri/youtube/raw_statistics_reference_data**" which is in json format.
Create an IAM role which named 'youtube-de-glue-s3-rule' and add two permission policies **AmazonS3FullAccess** and **AWSGlueServiceRole**.
Add a Databases as the target database named 'YouTube-de-raw'. After that it's fine to review and create the Glue Crawler.
View the table schema in Tables

5. Use Athena and SQL to query Data

In this step I created an S3 bucket which is for Athena query result. However, when I try to run query I have an Error: **Row is not a valid JSON Object - JSONException: A JSONObject text must end with '}' at 2 [character 3 line 1]**
Because it has the column which is array in Json format, Athena cannot read it. In terms of Json file format. Everything need to be in one single line.
Solution: we need to trigger AWS Lambda function to transform JSON to Apache Parquet format.

6. Writing ETL job in lambda and cleaning data

1. Create a lambda function named 'youtube-de-raw-useast1-lambda-json-parquet' \
2. Create an IAM role named 'youtube-de-raw-useast1-lambda-role' and attach "AmazonS3FullAcces" permission policy to it
3. Use the IAM role named 'youtube-de-raw-useast1-lambda-role' in the lambda function and create function.
   Edit the function:

```
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']
    path_split = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'], encoding='utf-8').split('/')
    key = '/'.join(path_split[1:])
#    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
```

Create a test event and add a new s3 bucket for cleaned data. After testing, I get several errors. Here is the process I tried to fix the error.
1. First, I **add a Layer** : AWSSDKPandas-Python38
Version: 5
2.I add a new permission role 'AWSGlueServiceRole' to the s3 bucket and **increase the Timeout** in General configuration.
3. Attach the IAM Role **youtube-de-raw-useast1-lambda-role** with a new policy which is **AmazonS3FullAcces** policy
4. Pervious code still got error because the Handler name didn't match. Also the default runtime is Node.js not python so I change the Handler name to **index.lambda_handler** and runtime: **Python 3.8**

5. Create a new database for cleaned data version in Glue job called '**de_youtube_cleaned**'

After completing the above steps, my s3 bucket has clean parquet file format data and Glue automatically performs the ETL process to generate a new table and target database is db_yuotube_cleaned.
Later I realize when I create the event and edit the Event JSON. I put the S3 bucket name in the key area. I should put everything after the s3 bucket and delete the bucket name in key area.
After this lambda ETL job, I am able to query the new database using Athena.

6. Building a Glue Crawler

Building a new Glue Crawler for the s3 data source **s3://youtube-de-raw-useast1-dev/youtube/raw_statistics/** and use the same database **youtube-de-raw**, after we run this crawler, we will get a new partitioned table. Because we use different way to unload the data into the S3 bucket. The partition key is region. So, we can easily use Athena to query our data based on the region.
Trying to use Athena to join the raw table with cleaned table:

```
SELECT * FROM "AwsDataCatalog"."youtube-de-raw"."raw_statistics" a

INNER JOIN "db_youtube_cleaned"."cleaned_statistics_reference_data" b

ON a.category_id = b.id;
```

Got this error message: SYNTAX_ERROR: line 3:18: '=' cannot be applied to bigint, varchar
So I change the column type to integer

```
SELECT a.title, a.category_id, b.snippet_title

FROM "AwsDataCatalog"."youtube-de-raw"."raw_statistics" a

INNER JOIN "db_youtube_cleaned"."cleaned_statistics_reference_data" b

ON a.category_id = cast(b.id as int)

WHERE a.region = 'ca'
```

Instead of this way to change the column type, I can also change the column type using table we created and edit data type in Glue using edit Schema. Which is called preprocessing.

7. Preprocessing:

1. First, go to the S3 bucket and delete the cleaned version of the data.
2. Go to the Glue table and edit the schema. Change the column id to bigint data type.
3. Go to lambda and create a new event to transform the raw_statistics_reference_data into parquet format.
Then I can query the data without cast.
More efficient for query because I don't need to do cast every time query the data
Next I would like to use Glue to create a job to change the schema of raw_statistics data into bigint data type.

8. Add a job in Glue:

pyspark job Script

```
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 = "youtube-de-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 = "youtube-de-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://youtube-de-raw-useast1-
dev/youtube/raw_statistics/"}, format = "parquet", transformation_ctx = "datasink4"]
## @return: datasink4
## @inputs: [frame = dropnullfields3]
```

```
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://youtube-de-raw-useast1-cleaned/youtube/raw_statistics/",

"partitionKeys": ["region"]}, format = "parquet", transformation_ctx = "datasink4")

job.commit()
```

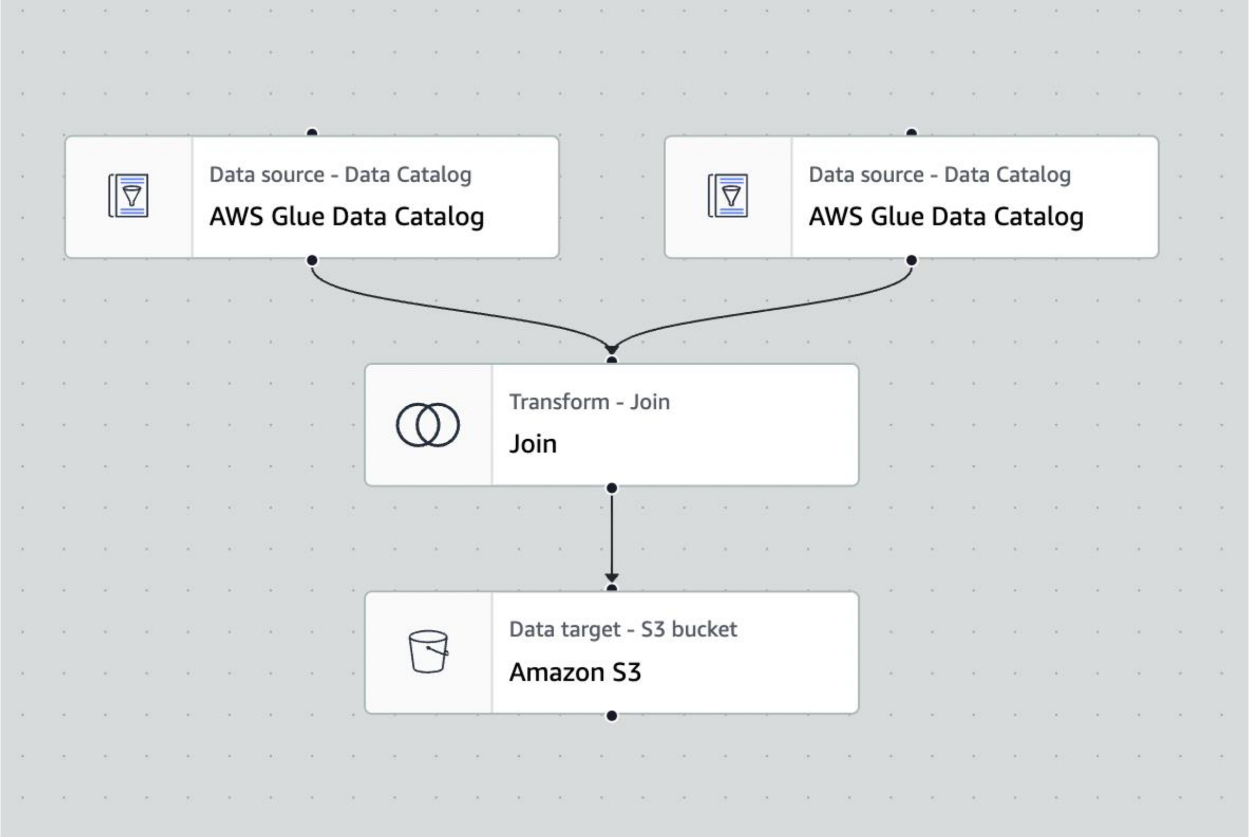
After transferring raw_statistics into parquet format. Create a crawler to determine the schema of the data, and then creates metadata tables in the db_youtube_cleaned database

8. After transferring raw_statistics into parquet format. Create a crawler to determine the schema of the data, and then creates metadata tables in the db_youtube_cleaned database.

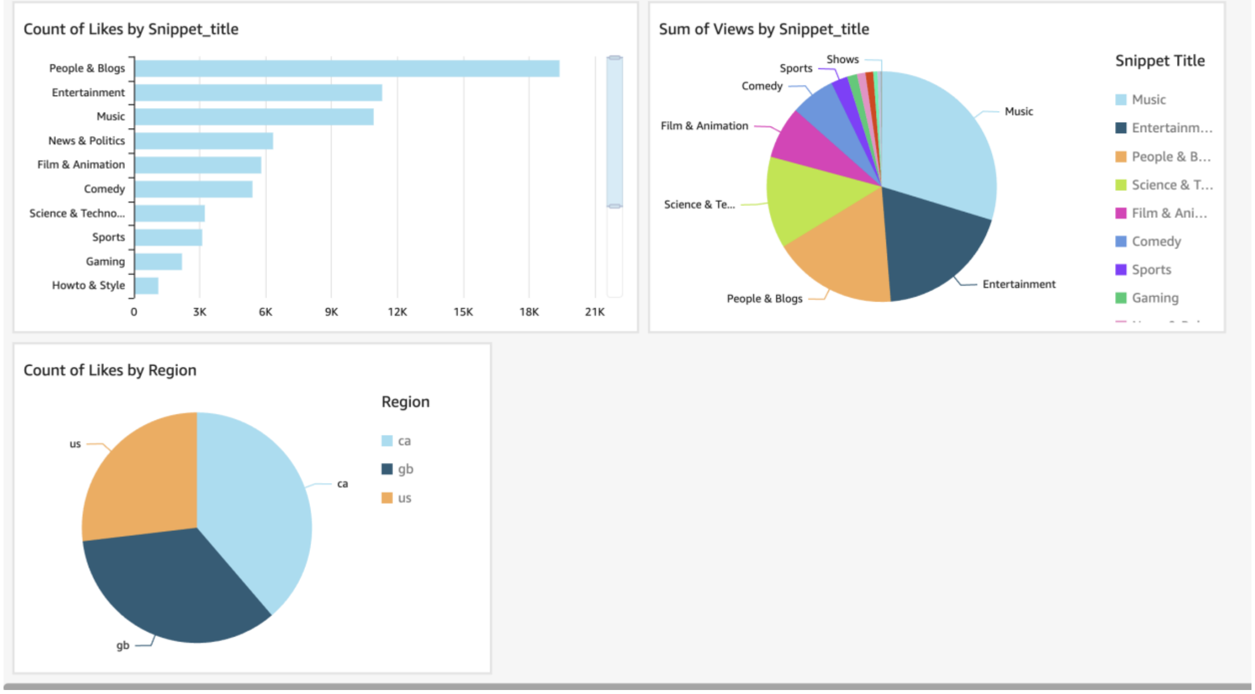
Since Before I only test one region to transform from json to parquet. Now I am going to using Lambda to create a ETL job trigger transformations every time a new file is being uploaded into S3 buckets.
1.Add a trigger in my lambda function
trigger source: S3
Bucket: s3://youtube-de-raw-useast1-dev
Event type: All object create events
Prefix: youtube/raw_statistics_reference_data/
Suffix: .json
2.delete all json file in my S3 bucket raw and cleaned version and upload all files using AWS CLI.
3.lambda function will automatically convert all json file into parquet format and stored in s3 cleaned version bucket.
After that I have cleaned format of all files in the database db_youtube_cleaned and the table name is raw_statistics and cleaned_statistics_reference_data

9. Building ETL Pipeline in Glue

I will use AWS Glue to add a job combine two cleaned version of data together and stored in a new S3 bucket for analysis.



10.AWS QUICKSIGHT SETUP:
Visualization



Count of Likes by Snippet_title

Sum of Views by Snippet_title

Count of Likes by Region