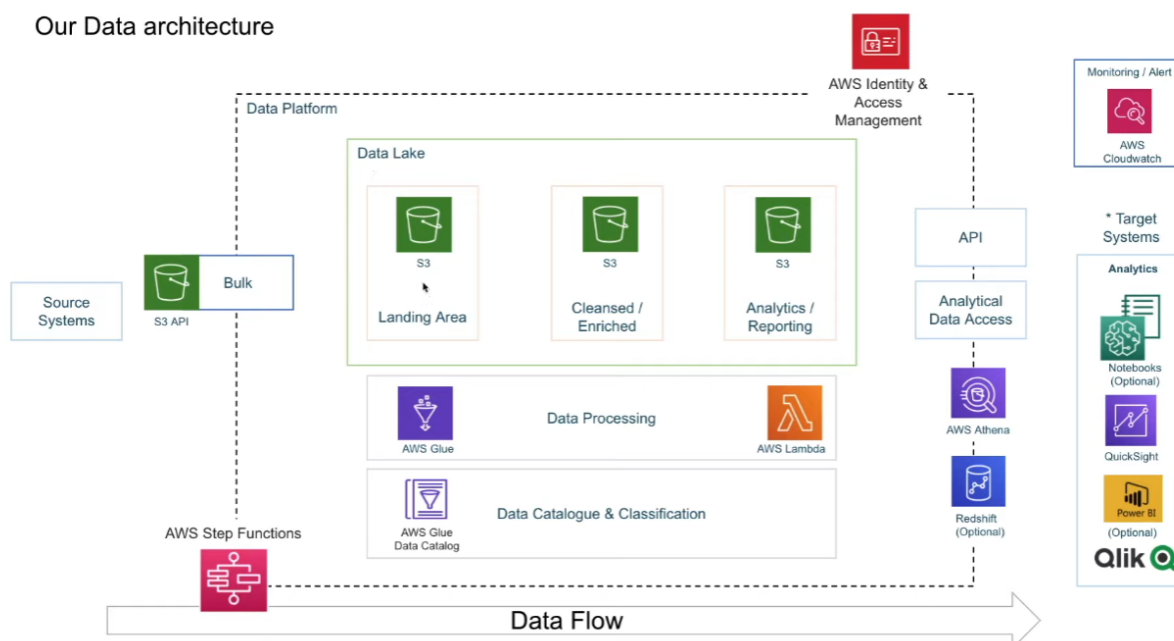


Youtube Data Analysis

1. Project Architecture.

Our Data architecture



2. Information about Dataset.

- Top Trending Videos.
- What is Trending?
 - YouTube uses factors, including users interactions e.g. number of views, shares, comments and likes.
 - Not the most-viewed videos overall for the calendar year
- Source: Kaggle; Data Collected using Youtube API
<https://www.kaggle.com/datasnaek/youtube-new>
- Json File has each category attached in Data and CSV file has multiple video metadata partitioned by region.
- The data also includes a `category_id` field, which varies between regions. To retrieve the categories for a specific video, find it in the associated JSON.

3. Interaction with AWS

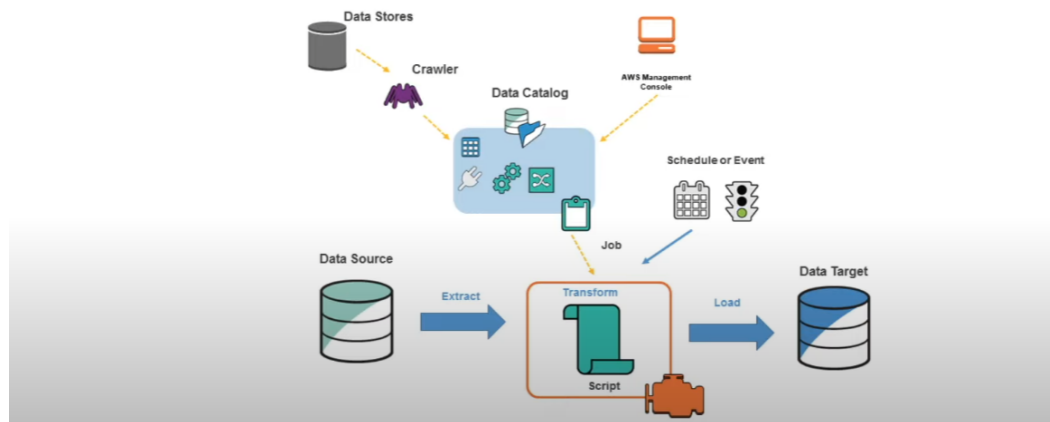
- a. Used AWS CLI : To Upload Data in new created S3 Bucket (de-on-youtube-raw-useast1-dev14) Both JSON and CSV

```
upload: .\RU_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/RU_category_id.json
PS E:\youtube> aws s3 cp . s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/ --recursive --exclude "*" --include "*.json"
>>
upload: .\RU_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/RU_category_id.json
upload: .\CA_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/CA_category_id.json
upload: .\MX_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/MX_category_id.json
upload: .\US_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/US_category_id.json
upload: .\IN_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/IN_category_id.json
upload: .\GB_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/GB_category_id.json
upload: .\DE_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/DE_category_id.json
upload: .\JP_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/JP_category_id.json
upload: .\FR_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/FR_category_id.json
upload: .\KR_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/KR_category_id.json
PS E:\youtube>
```

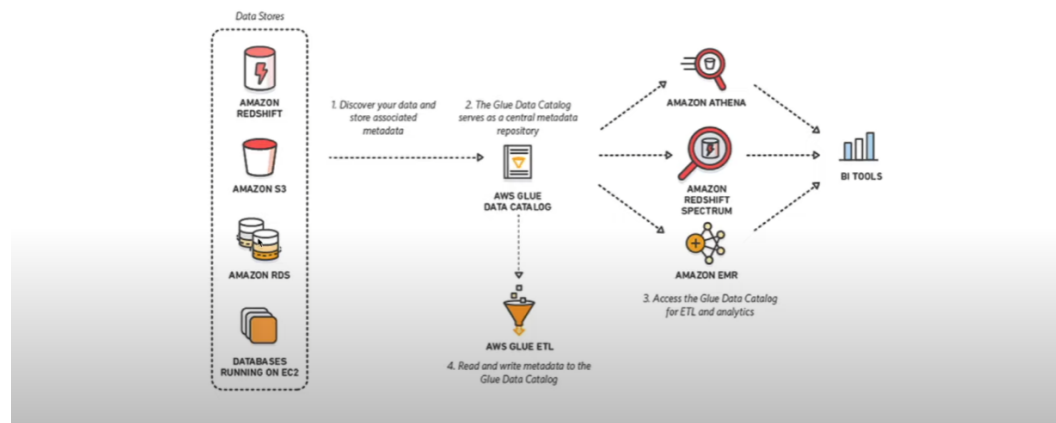
4. Created AWS Glue Catalog

- a. What is AWS Catalog

What is the AWS Glue Catalog



- b. How AWS Glue Catalog Works



c. Created Role in IAM for Glue to access S3 in Crawler.

<input type="checkbox"/>	Policy name ↗
<input type="checkbox"/>	+ AmazonS3FullAccess
<input type="checkbox"/>	+ AWSGlueServiceRole
<input type="checkbox"/>	+ AWSGlueConsoleFullAccess

d. Created a crawler to get metadata(Information about the table) in the raw dataset.

Crawler properties

Name
de-on-youtube-raw-glue-catalog-1

IAM role
[s3-glue-role ↗](#)

Description

-

Security configuration

-

e. Tested Data in Athena

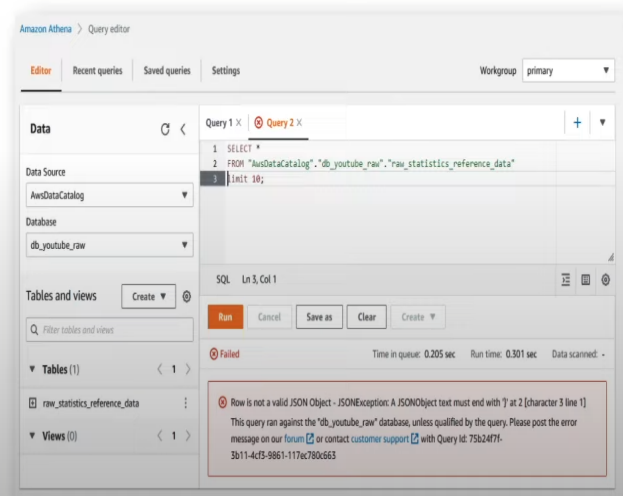
- Created S3 Bucket to Store Athena Output.
- Found Error in JSON Data Querying. (We only need Items from JSON).

We only need this nested Key!

```
{
  "kind": "youtube#videoCategoryListResponse",
  "etag": "\"ld9b1NPKjA9gjV7EZ4EkeEGrhao/1v2mrzYSY66onNlt2Qj33hkQ2k\"",
  "items": [
    {
      "kind": "youtube#videoCategory",
      "etag": "\"ld9b1NPKjA9gjV7EZ4EkeEGrhao/Xy1nB4_yLrHy_8nKmPBggty2nZQ\"",
      "id": "1",
      "snippet": {
        "channelId": "UCBR8-68-B28hp28mDPdntcQ",
        "title": "Film & Animation",
        "assignable": true
      }
    },
    {
      "kind": "youtube#videoCategory",
      "etag": "\"ld9b1NPKjA9gjV7EZ4EkeEGrhao/UZ1oLI22oxIn045Z7FR3a3Ny7A\"",
      "id": "2",
      "snippet": {
        "channelId": "UCBR8-68-B28hp28mDPdntcQ",
        "title": "Autos & Vehicles",
        "assignable": true
      }
    }
  ]
}
```

CA_category_id.json

iii.



- Why?:** JSON records in data files must appear one per line, an empty file would produce a NULL record. This is because Hadoop partitions files as text using CR/LF as a separator to distribute work.

☰ README.md

JSON Data Files

Upload JSON files to HDFS with `hadoop fs -put` or `LOAD DATA LOCAL`. JSON records in data files must appear *one per line*, an empty line would produce a NULL record. This is because Hadoop partitions files as text using CR/LF as a separator to distribute work.

The following example will work.

```
{ "key" : 10 }
{ "key" : 20 }
```

➡ The following example will not work.

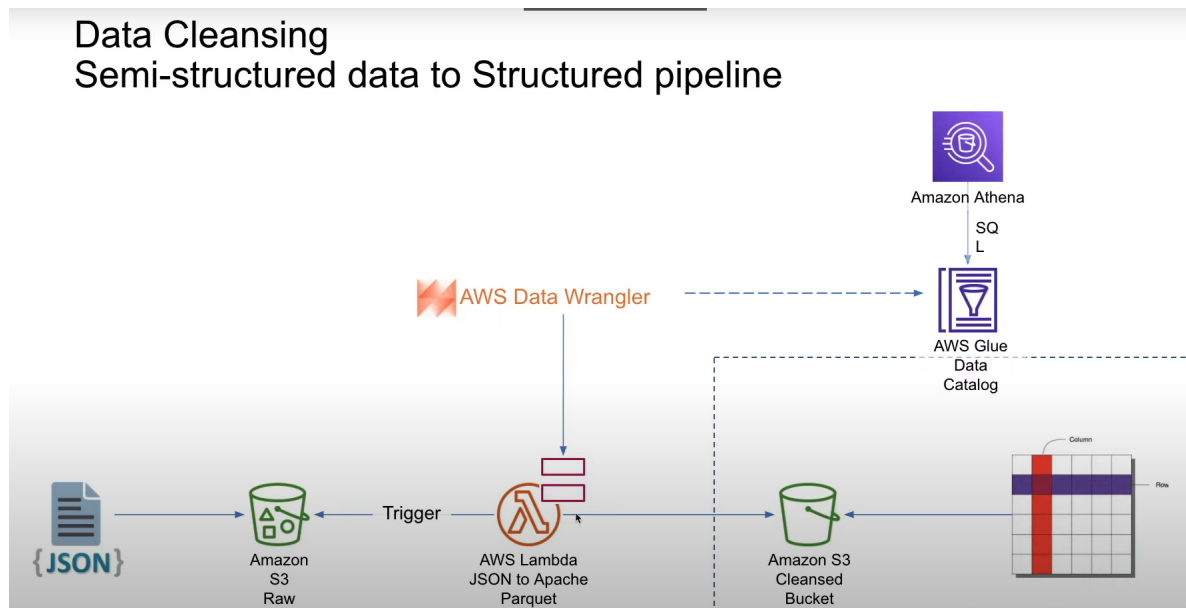
```
{
  "key" : 10
}
{
  "key" : 20
}
```

g. **Goal:** Data Cleansing

Create our light ETL: JSON to Apache Parquet.



- **Architecture for Data Cleaning semi structured to structured Pipeline using Lambda**



- **Created Spark code for Transformation in Lambda using AWS Wrangler.**

```

1  import awswrangler as wr
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(f's3://{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

```

5. Why we chose Lambda for Transformation?

- One of the AWS compute services
- Serverless
- High Available and scalable
- Limits at this moment:
 - Deployment Package is 50 MB
 - 10 GB for Memory
 - 6 vCPUs
 - 15 min Timeout

Comparing the different data storage options



This table compares the characteristics of these four different data storage options for Lambda:

	Amazon S3	/tmp	Lambda Layers	Amazon EFS
Maximum size	Elastic	512 MB	50 MB (direct upload; larger if from S3).	Elastic
Persistence	Durable	Ephemeral	Durable	Durable
Content	Dynamic	Dynamic	Static	Dynamic
Storage type	Object	File system	Archive	File system
Lambda event source integration	Native	N/A	N/A	N/A
Operations supported	Atomic with versioning	Any file system operation	Immutable	Any file system operation
Object tagging	Y	N	N	N
Object metadata	Y	N	N	N
Pricing model	Storage + requests + data transfer	Included in Lambda	Included in Lambda	Storage + data transfer + throughput
Sharing/permissions model	IAM	Function-only	IAM	IAM + NFS
Source for AWS Glue	Y	N	N	N
Source for Amazon QuickSight	Y	N	N	N
Relative data access speed from Lambda	Fast	Fastest	Fastest	Very fast

- Querying our cleaned Data in Athena .

Query 5 × Query 6 ×

```
1 SELECT * FROM "AwsDataCatalog"."db_youtube_cleaned"."cleaned_statistics_reference_data" limit 10;
```

SQL Ln 1, Col 1

Run again Cancel Save Clear Create

Completed Time in queue: 0.205 sec Run time: 0.657 sec Data scanned: 2.15 KB

Results (10) Copy Download results

Search rows

#	kind	etag	id	snippet_channelid	snippet_title	snippet_assignable
1	youtube#videoCategory	"mZyskBQFythfE4IrbTleOgYYfBU/Xy1mB4_yLrHy_8mKmpBggy2mZQ"	1	UCBR8-60-828hp2BmDPdntcQ	Film & Animation	true
2	youtube#videoCategory	"mZyskBQFythfE4IrbTleOgYYfBU/UJ1oLilz2dxhO45ZTFR3a3NyTA"	2	UCBR8-60-828hp2BmDPdntcQ	Autos & Vehicles	true

6. Joining JSON and CSV Data.

```
SELECT * FROM
  "db_youtube_cleaned"."raw_statistics" a
  INNER JOIN
  "db_youtube_cleaned"."cleaned_statistics_reference_data" b
  ON a.category_id=b.id
;
```


Query 8 : X Query 9 : X

```

1 SELECT *
2 FROM "db_youtube_cleaned"."raw_statistics" a
3 inner join "db_youtube_cleaned"."cleaned_statistics_reference_data" b ON a.category_id = b.id
4 where region = 'ca';

```

SQL Ln 3, Col 19

Run Explain Cancel Clear Create

Completed Time in queue: 141 ms Run time: 2.012 sec Data scanned: 603.32 KB

Results (100+)

Search rows

#	video_id	trending_date	title	channel_title	category_id
1	iyewTFLzNZ4	18.16.01	Frog The Rooster-Savannah is home!	Frog The Rooster	22
2	iyewTFLzNZ4	18.16.01	Frog The Rooster-Savannah is home!	Frog The Rooster	22

7. Changed Data Type of Parquet file output for efficient running of code.

Steps: 1. Keep the data type change in the data catalogue

tables > cleaned_statistics_reference_data

Add column

	Column name
1	kind
2	etag
3	id
4	snippet_channelid
5	snippet_title
6	snippet_assignable

String type

Column type

string

array

bigint

binary

boolean

Update

Comment

Last updated 27

2. Delete our testing JSON file
3. Confirm APPEND in Lambda
4. Run Test event in Lambda

Event JSON

Format JSON

```
13      "sourceIPAddress": "127.0.0.1"
14    },
15    "responseElements": {
16      "x-amz-request-id": "EXAMPLE123456789",
17      "x-amz-id-2": "EXAMPLE123/5678abcdefghijklmbdaisawesom/mnopqrstuvwxyzABCDEFGH"
18    },
19    "s3": {
20      "s3SchemaVersion": "1.0",
21      "configurationId": "testConfigRule",
22      "bucket": {
23        "name": "de-on-youtube-raw-useast1-dev",
24        "ownerIdentity": {
25          "principalId": "EXAMPLE"
26        },
27        "arn": "arn:aws:s3:::de-on-youtube-raw-useast1-dev"
28      },
29      "object": {
30        "key": "test%2Fkey",
31        "size": 1024,
32        "eTag": "0123456789abcdef0123456789abcdef",
33        "sequencer": "0A1B2C3D4E5F678901"
34      }
35    }
36  }
37 ]
38
```

5. Copy again our data, from our laptops (AWS CLI)

```
upload: \RU_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/RU_category_
id.json
PS E:\youtube> aws s3 cp . s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/ --recursive --exc
lude "*" --include "*.json"
>>
upload: \RU_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/RU_category_
id.json
upload: \CA_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/CA_category_
id.json
upload: \MX_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/MX_category_
id.json
upload: \US_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/US_category_
id.json
upload: \IN_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/IN_category_
id.json
upload: \GB_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/GB_category_
id.json
upload: \DE_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/DE_category_
id.json
upload: \JP_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/JP_category_
id.json
upload: \FR_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/FR_category_
id.json
upload: \KR_category_id.json to s3://de-on-youtube-raw-useast1-dev14/youtube/raw_statistics_reference_data/KR_category_
id.json
PS E:\youtube>
```


6. Add the S3 Trigger to Lambda

Triggers (1) Info

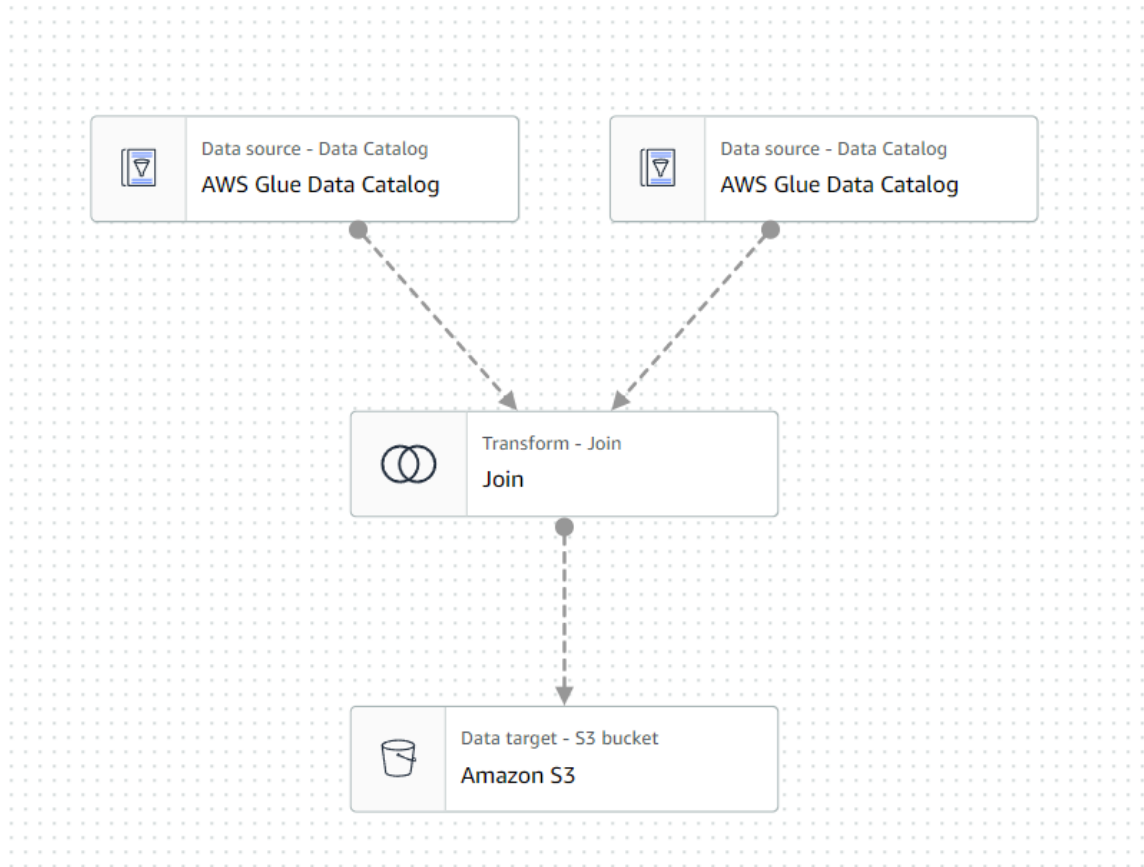
☐

Trigger

☐

 **S3: de-on-youtube-raw-useast1-dev14**
arn:aws:s3:::de-on-youtube-raw-useast1-dev14
[Details](#)

8. Created ETL Job for analytics



9. Build Reports based on Cleaned Data for Analysis

a. Cleansed vs Analytics Layer

Cleansed vs Analytics layer

Using Cleansed Layer



```
1 SELECT ref.snippet_title, stats.title, stats.title
2 FROM raw_statistics stats
3 INNER JOIN cleansed_statistics_reference_data ref on (stats.category_id = ref.id)
4 WHERE ref.id=2
```

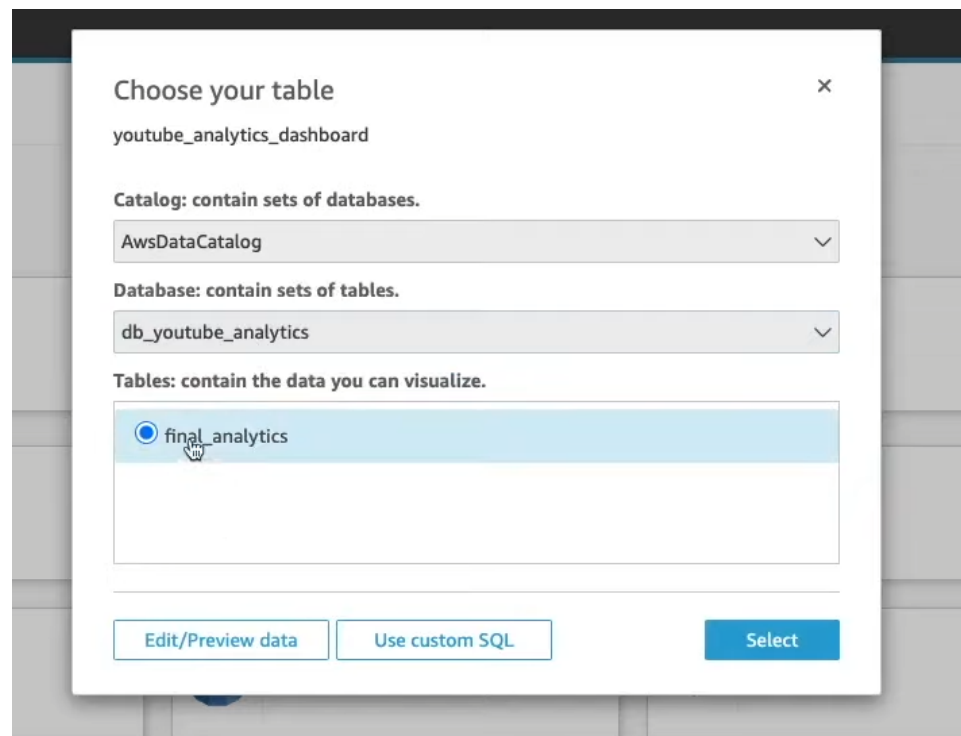
Using Reporting Layer



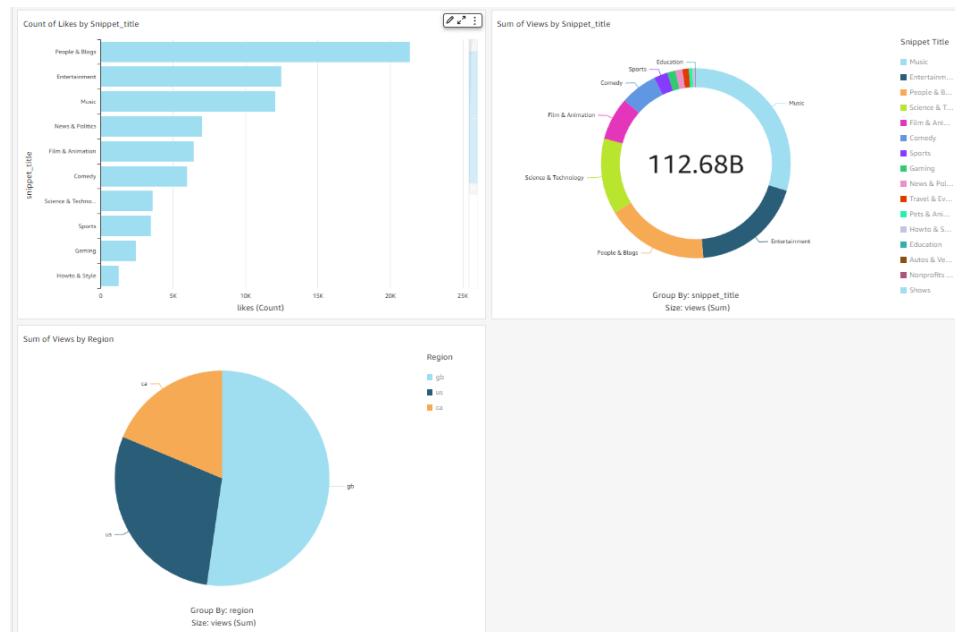
```
1 SELECT snippet_title, title, title
2 FROM rpt_youtube_statistics_categories
3 WHERE id=2
```

b. Dashboards

i. Connecting to Athena



ii. Building Reports



Learnings:

- The project involved extracting, cleaning, and transforming YouTube trending video data from around the world.
- Raw data was stored in an Amazon S3 bucket and cataloged using a Glue Crawler.
- Data cleaning and transformation was performed using Apache Spark.
- An ETL job was created using AWS Lambda to automate the data cleaning and transformation process.
- The cleaned data was queried using Amazon Athena and SQL to generate reports and insights.
- An ETL pipeline was created to copy the cleaned data to a separate S3 bucket for analysis.
- Amazon QuickSight was used to build various reports and visualizations based on the data.
- This project enabled the efficient and effective analysis of YouTube trending video data, resulting in the discovery of trends and patterns.