

Project Overview-

Welcome to my project.

I've created an End to end data pipeline project on a Spotify data set using AWS Glue and Amazon Crawler.

Then Used Amazon S3 bucket to store the input and output files accordingly and used Amazon Athena to query the tables made by the crawler.

The Goal of this project as shown in the architecture diagram, is to demonstrate the features of the aws platform to upload documents, extract and clean the documents and transform them according to the user's needs.

Under this section, I've mentioned each and every steps by which one can be able to visualize and understand the scenario, the objectives and the tools applied while making this project from root level.

Thank you.

Project Architecture-

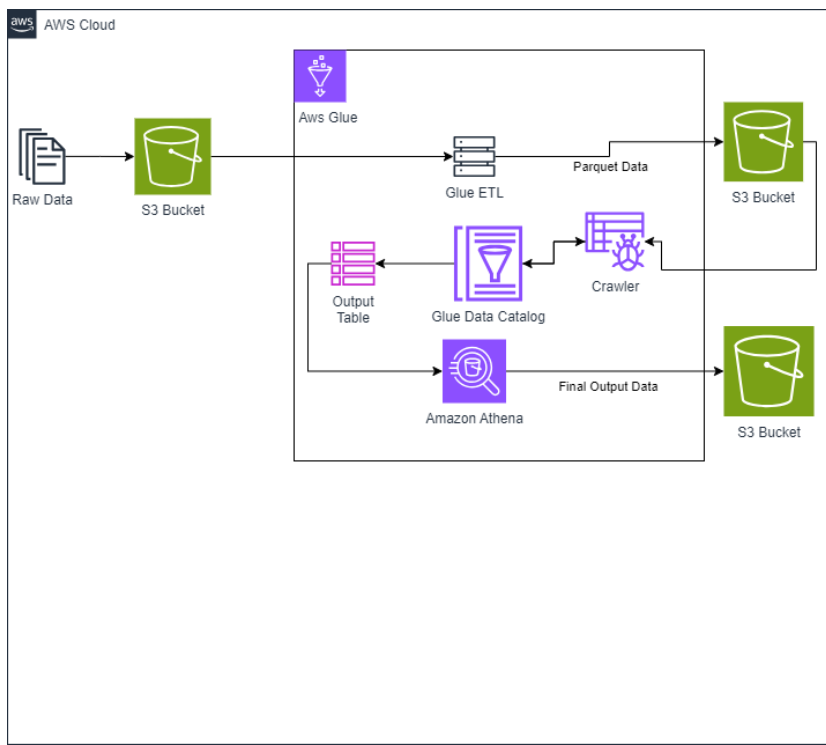


Fig1-Project Architecture

Detailed steps-

1. Make an Amazon AWS account first and validate your account to keep using it (It provides a one year of free tier service which one can use efficiently for making projects or in their day to day activities.)

After that, you can use it as “Pay-as-you-go” model by which, you will have to pay for the services that you use only.

2. **Collecting dataset-** Collecting random datasets can be a very easy task as we have a lot of datasets that are available in the internet for free. You can choose any datasets of your choice and start to work on them.

For this particular project, I’ve used the Spotify (2013) dataset which I’m providing within this folder.

3. **Uploading the file-** At first make a S3 bucket and after the bucket creation, go to the S3 bucket and upload the csv files.

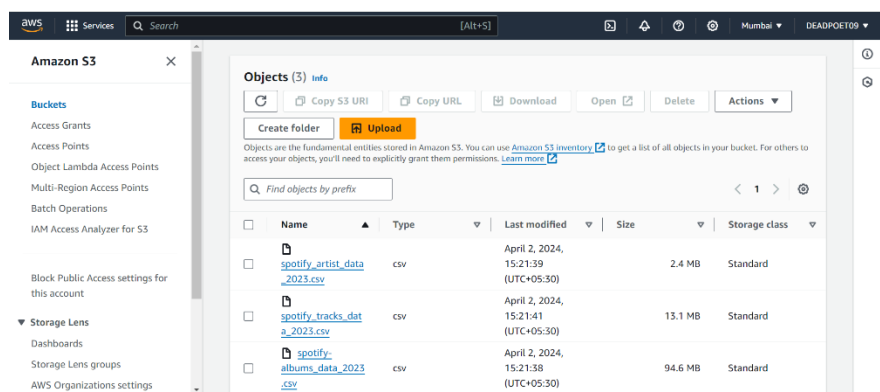


Fig 2-Input Files

4. After uploading the files, go to the Amazon Glue (It is not a free tier service but its cost is very minimal (For one hour of use, it will cost you under 100 INR) and start building the ETL pipeline.

5. **ETL pipeline-i)** At first, create a data source as S3 bucket where you stored your data sets.

ii) From the transform section, pull the join function to put a join between “Album” and the “Artist” datasets and name it a specific value let’s say “Join-Album-Artist”. (Put the format according to the dataset. Here we are using CSV so I made that as CSV)

iii) Again pull one more join function to join the “Tracks” and “Join-Album-Artist” values and rename it with “Join_With_Tracks”. (Put the format according to the dataset. Here we are using CSV so I made that as CSV)

iv) After doing this, we can be able to find out that there are many similar columns (For example, the “id” column is repeated twice). So we need a Transform function to delete all the unnecessary data from the “Join_With_Tracks” dataset.

v) Finally you can bring an output bucket to put the output and mark it as a parquet format and save it(**The reason to use parquet format-Less data scanned at ingestion leading to faster scans and lower query and memory costs because you only have to read the columns that you actually need.**).

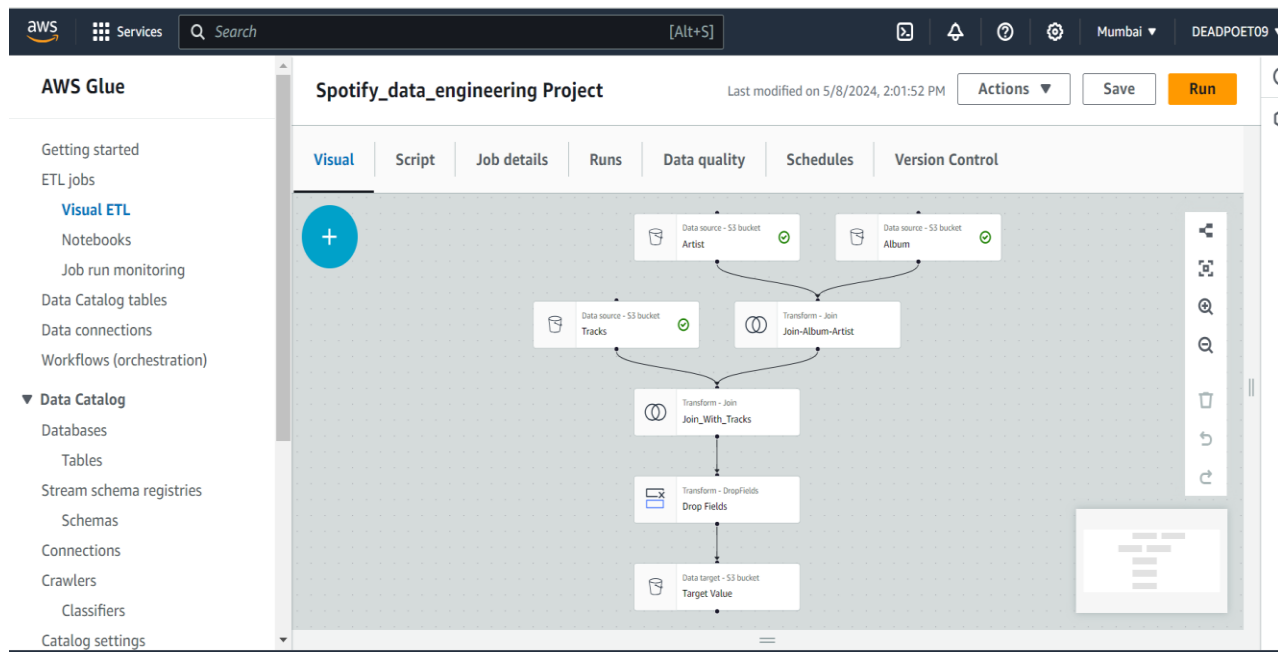


Fig3- The ETL process

vi) After that just hit Run and you are good to go (Provided with the “Spotify_data_engineering Project.py” that was auto generated after the “Run” button was executed).

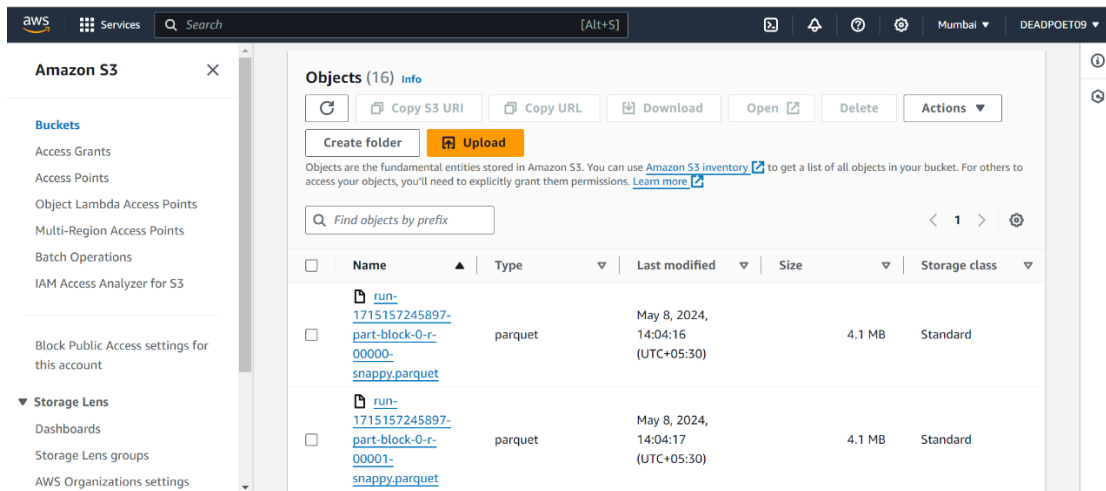


Fig4-Datas after the ETL process (Parquet format)

6. Configuring Database- Make a database of any name on which we can be able to store the Tables. (Use proper IAM rules to For Glue to access the Database).

7. Create a Glue crawler to process the parquet datas to create a data-catalog which can be further used for meaningful information.

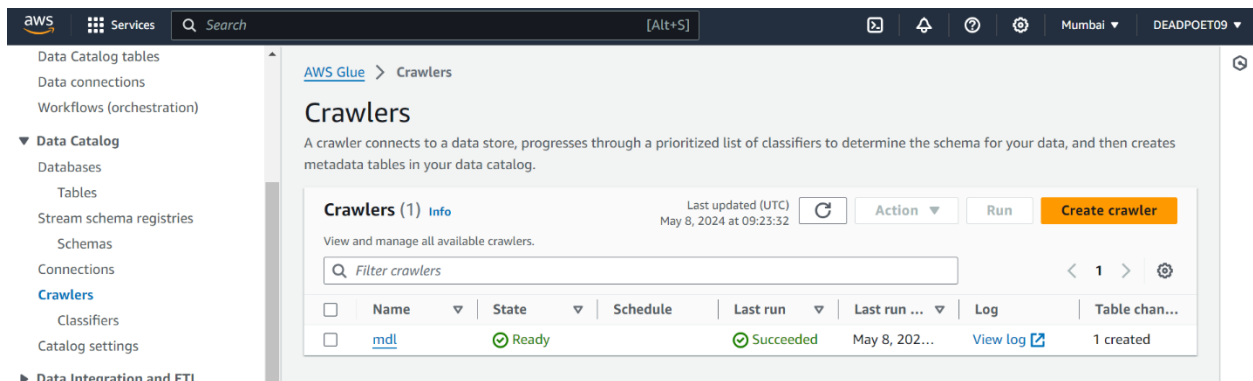


Fig5-Crawler

The screenshot shows the AWS Glue console interface. On the left is a navigation menu with options like 'Getting started', 'ETL jobs', 'Data Catalog tables', and 'Data Catalog'. The main panel displays the 'Schema' tab for a table named 'csv1_output'. It shows a list of columns with their names, data types, and whether they are partition keys.

#	Column name	Data type	Partition key	Comment
1	followers	string	-	-
2	artist_10	string	-	-
3	track_id	string	-	-
4	artist_1	string	-	-
5	artist_popularity	string	-	-
6	artist_4	string	-	-
7	artist_id	string	-	-
8	track_number	string	-	-

Fig6-Table after crawler is executed

8.Finally use The Amazon Athena to query the table accordingly to your need.

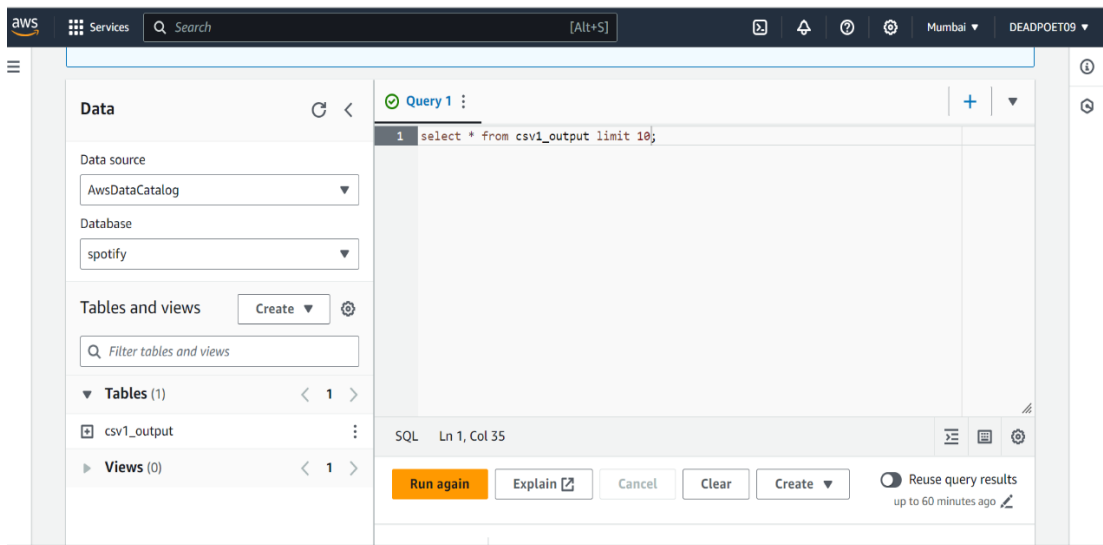


Fig7-Run query on Amazon Athena

Results (10)

Search rows

#	followers	artist_10	track_id	artist_1	artist_po
1	2316620		4nUvJlOecb3PwRvwzLXbAT		0
2	2847		2qLBTJNTF3S6TVkqBaU4HX		25
3	15		3YwNNG8RotsWyDlpK6TjF4		0
4	722		1G40or8eGQ3PO16A5tu570		2
5	49		4WqQxABa4q8WiWSOM0lpYE	Themis Georgoudis	1
6	550280		6MwjegGg50MTxjK3e0jrf		48
7	22		5lnAKpBpdFNAs1RkXDQ7g3		0
8	135		5DKohcffJEpi6uTaPOFXoG	Fntwxy	2

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Fig8-Athena Output

Amazon S3

Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3

Block Public Access settings for this account

▼ Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings

Objects (4) Info

Refresh Copy S3 URI Copy URL Download Open Delete Actions

Create folder Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	51784038-bfea-4443-ba00-ad7de09b8dd0.csv	csv	May 8, 2024, 14:37:18 (UTC+05:30)	4.0 KB	Standard
<input type="checkbox"/>	51784038-bfea-4443-ba00-ad7de09b8dd0.csv.metadata	metadata	May 8, 2024, 14:37:19 (UTC+05:30)	2.0 KB	Standard

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Fig9- Final Output in S3 Bucket