Spotify Playlist ETL Pipeline on AWS

Objective

The main objective of this project is to design and implement a **serverless ETL pipeline** that automates the collection, transformation, and analysis of Spotify playlist data. The goal is to enable efficient storage, schema inference, and analytical querying on playlist data without manual intervention.

Specifically, the pipeline aims to:

- **Automate data ingestion** from Spotify API on a weekly basis.
- **Transform raw JSON data** into structured, query-ready formats.
- **Leverage AWS services** (Lambda, S3, Glue, Athena) for scalability and cost-efficiency.
- **Enable SQL-based analytics** on playlist data for insights into tracks, artists, and genres.

Introduction

The Spotify API, specifically the Spotify Web API, is a RESTful API that allows developers to interact with Spotify's vast data catalog and services. It provides a programmatic interface to access and manage various aspects of Spotify.

Once upon a time, a music lover's Spotify playlist grew so big that keeping track of songs, artists, and trends became overwhelming. Manual tracking took hours, and insights were nearly impossible to get. That's why we need an ETL pipeline—to automatically extract, clean, and load playlist data into reports and dashboards, turning chaos into clear insights.

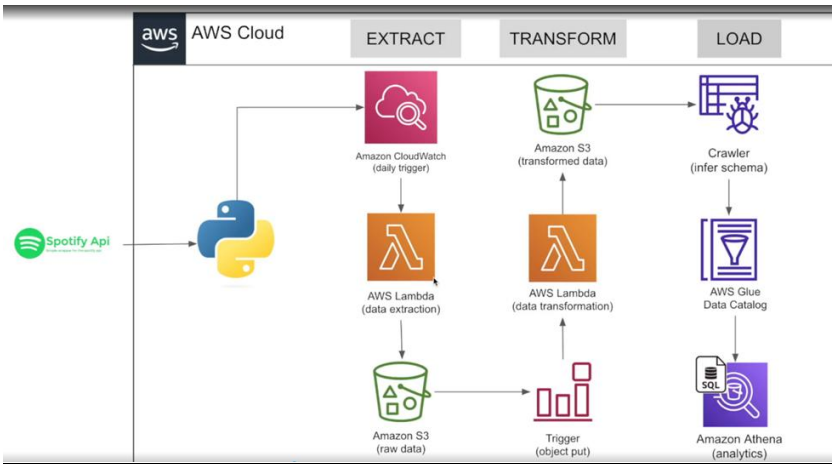
The goal of this project is to build a fully automated ETL pipeline for Spotify playlist data using AWS services. The pipeline is designed to:

- Extract playlist data from the Spotify API on a scheduled basis.
- Transform the raw JSON into structured, analysis-ready data.
- Load the transformed data into Amazon S3, with schema management in AWS Glue, for easy SQL querying through Amazon Athena.

By achieving this, the project enables seamless weekly updates, automated schema inference, and powerful analytics on Spotify playlists without manual intervention.

Architecture & Workflow

Architecture



Amazon CloudWatch:

Amazon CloudWatch is a monitoring and observability service that collects and tracks metrics, collects and monitors log files, and sets alarm to notify you of certain conditions. It provides a unified view of your resources and applications running on AWS and on-premises, helping you monitor performance, detect anomalies, troubleshoot issues, and automate responses. Key features include CloudWatch Logs for log analysis, CloudWatch Metrics for performance data visualization, alarms to trigger automated actions, and Container Insights for containerized applications.

AWS Lambda:

AWS lambda is a serverless, even-driven compute service provided by amazon web services (AWS). It allows users to run code without provisioning or managing servers. Users only pay for the compute time consumed when their code is executed.

Simple Storage Service (S3):

Amazon S3, or Amazon Simple Storage Service, is an object storage service provided by Amazon Web Services (AWS). It is designed to offer industry-leading scalability, data availability, security, and performance for storing and retrieving any amount of data from anywhere on the web

AWS Crawler:

AWS Glue Crawler is a feature of AWS Glue that automatically scans data repositories like S3, DynamoDB, and on-premises systems, extracts metadata and schema information, and populates the AWS Glue Data Catalog.

Amazon Athena:

Amazon Athena is a serverless, interactive query service provided by Amazon Web Services (AWS) that enables users to analyze data directly in Amazon Simple Storage Service (Amazon S3) using standard SQL.

Workflow

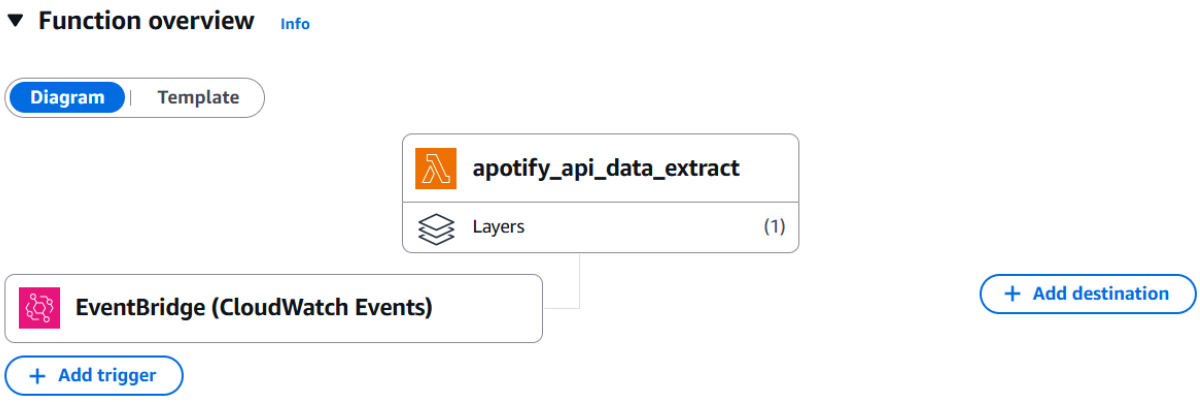
Data Extraction:

1. Amazon watch will trigger the AWS Lambda contains the data extraction code, which basically extracts data from Spotify API and then store the extracted raw data into S3 as a JSON format.

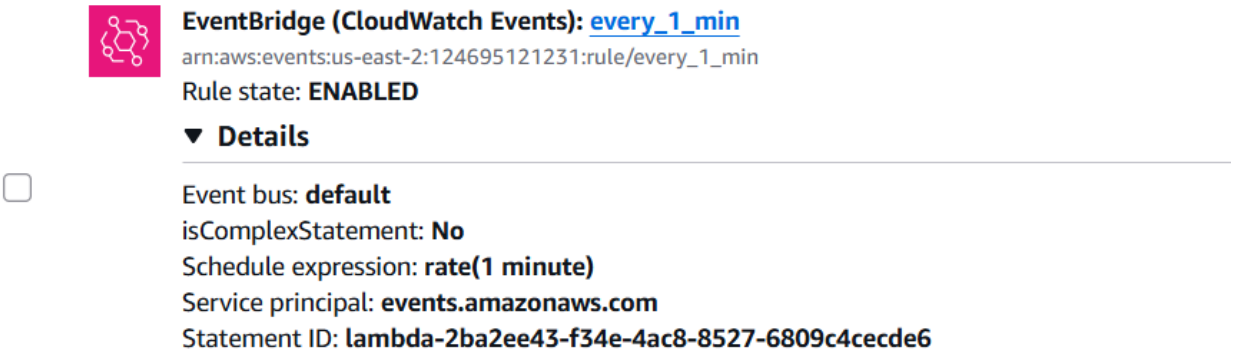
Code:

```python
import json
import os
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
import boto3
from datetime import datetime

def lambda_handler(event, context):

    cilent_id = os.environ.get('client_id')
    client_secret = os.environ.get('client_secret')

    client_credentials_manager = SpotifyClientCredentials(client_id=cilent_id, client_secret=client_secret)
    sp = spotipy.Spotify(client_credentials_manager = client_credentials_manager)
    playlists = sp.user_playlists('spotify')

    playlist_link = "https://open.spotify.com/playlist/2SM6rniZl84fEyMCBSKMQ8"
    playlist_URI = playlist_link.split("/")[-1].split("?")[0]

    spotify_data = sp.playlist_tracks(playlist_URI)

    cilent = boto3.client('s3')

    filename = "spotify_raw_" + str(datetime.now()) + ".json"

    cilent.put_object(
        Bucket="spotify-etl-thrivikram",
        Key="raw_data/to_processed/" + filename,
        Body=json.dumps(spotify_data)
```

2. EventBridge (CloudWatch Events) will trigger the Lambda function which has extract code. This EventBridge will execute the program every week or every month. The below figure is the diagrammatic representation of the trigger connected to Data Extract



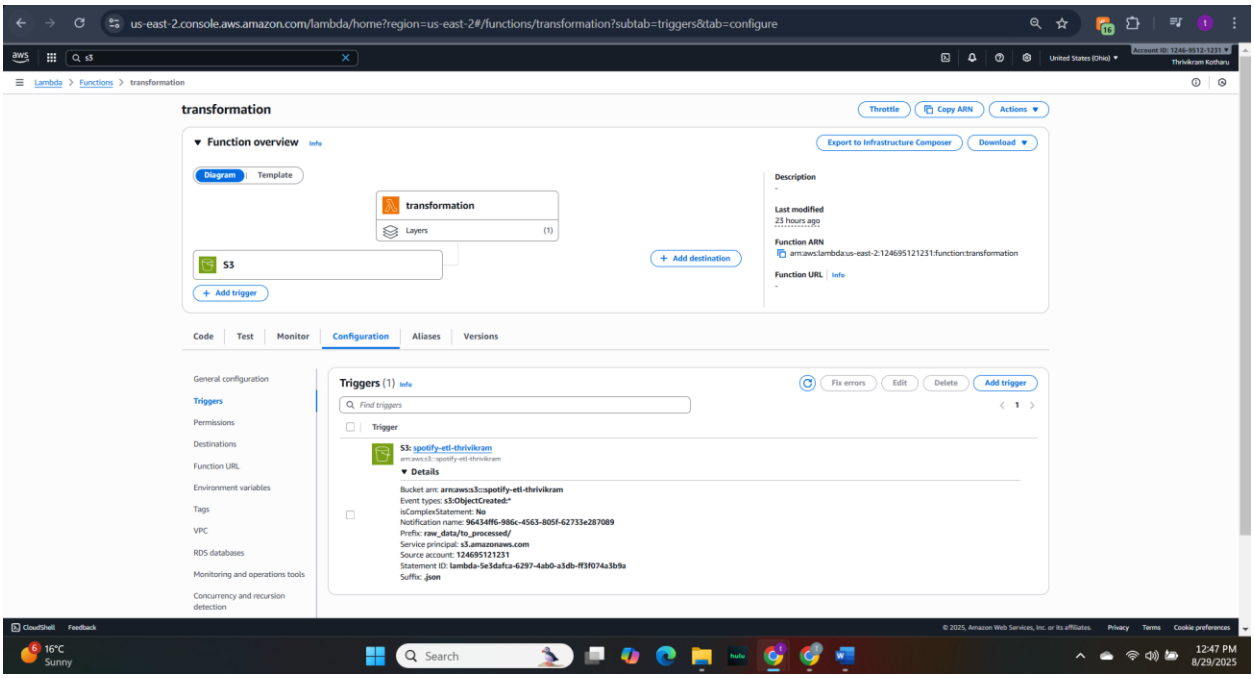
3. Below is the configuration of the same trigger. For testing purpose this trigger is set to trigger everyone minute but in real time this won't be necessary.



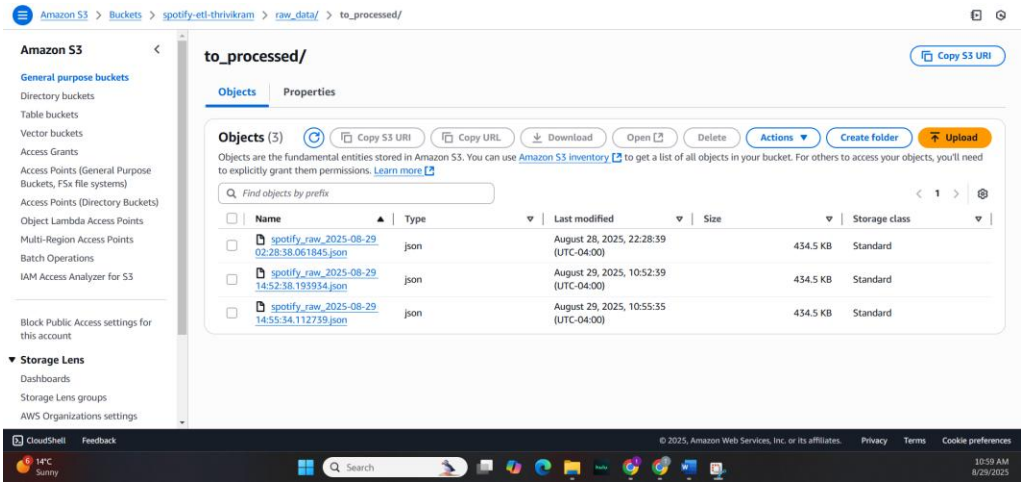
Data Transformation and Loading:

4. Now, Once this is triggered the extracted data will be staged in S3, the main reason to use it this way is because I am using trial version of lambda for this project which has very little RAM, to perform both Extract and transform together, so I am using the same S3 bucket but a different file to store both to_processed data(extracted data) and processed data(transformed data).

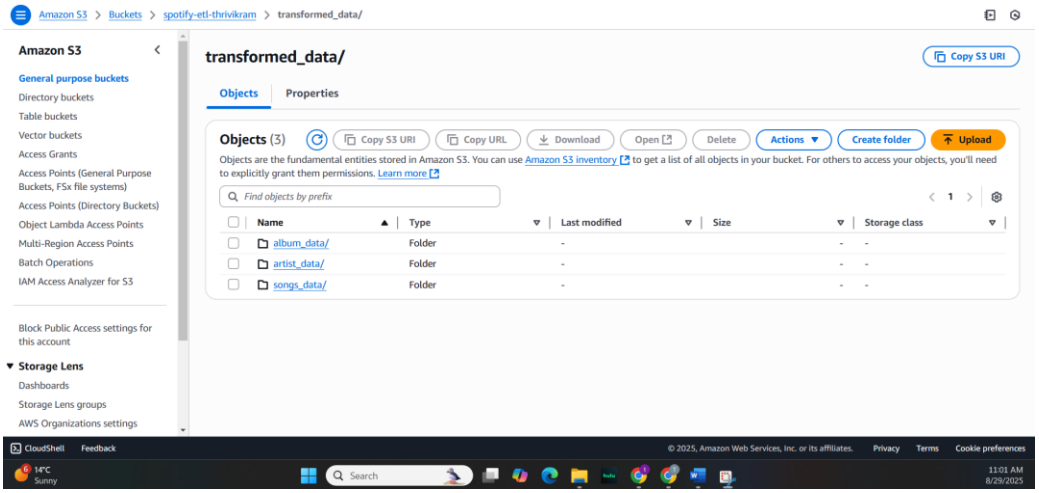
5. So, for transformation we will use a separate lambda function in AWS, we are going to trigger this lambda function once there is some new data or files in the to_processed folder in S3. Below is a screenshot of code and the diagrammatic representation of trigger and its configuration.

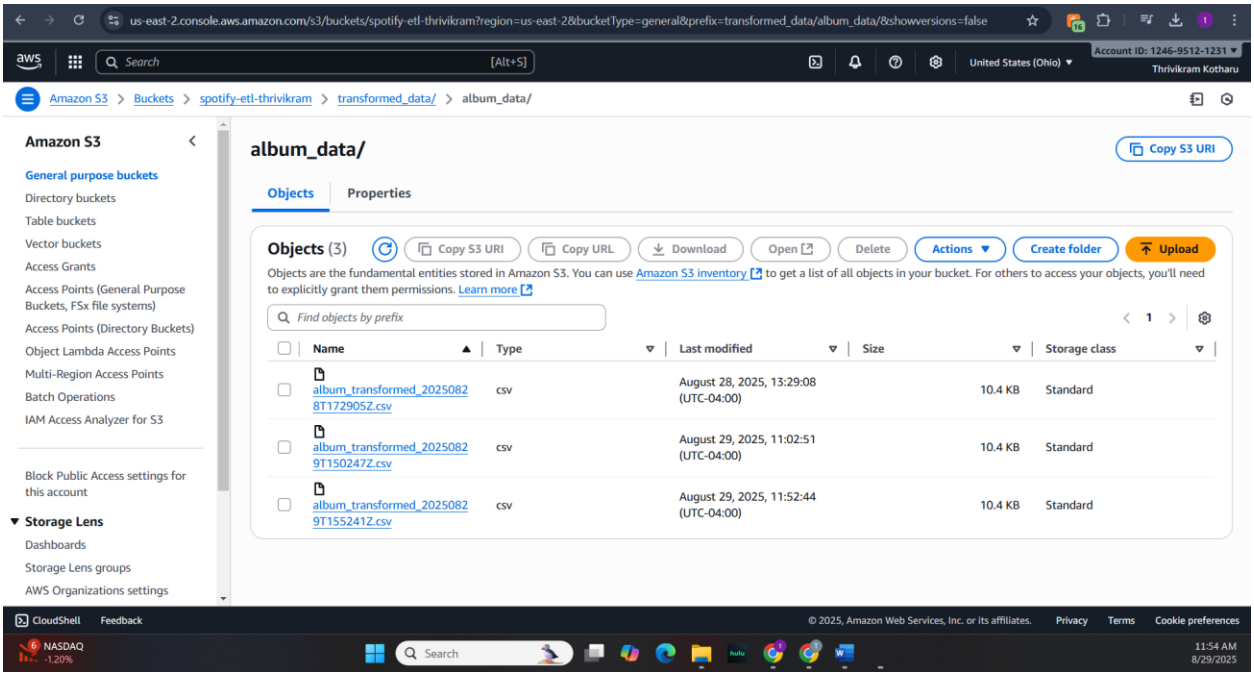


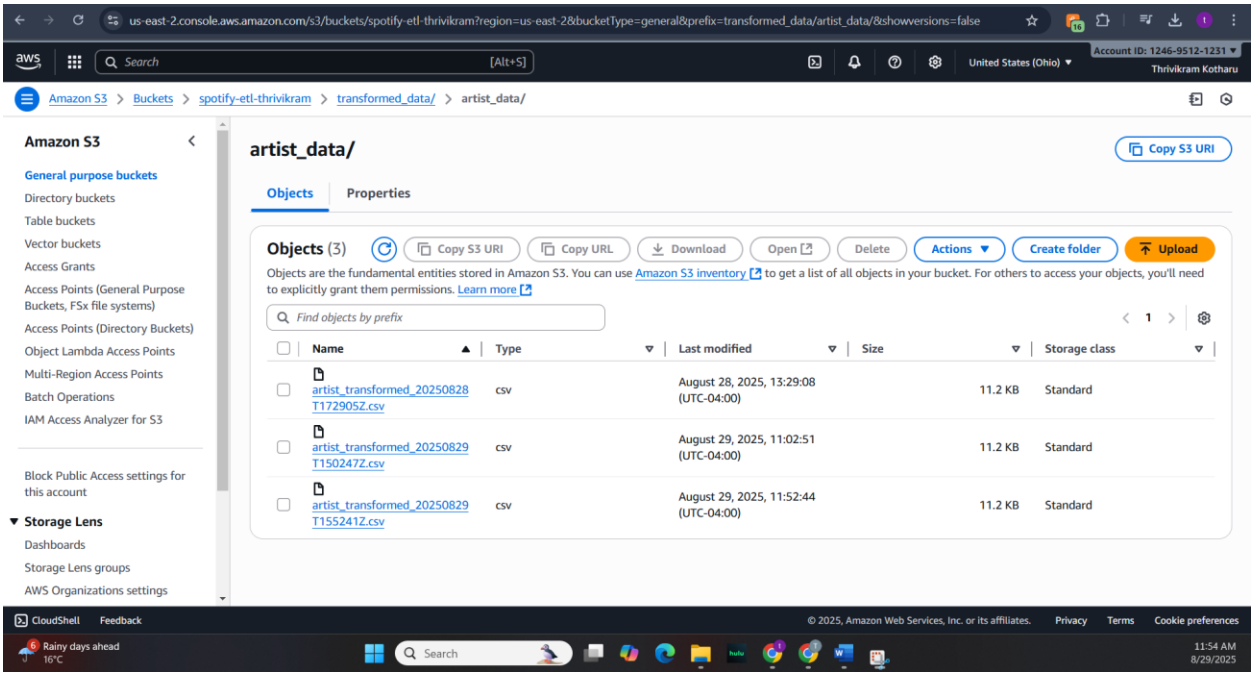
to_processed data

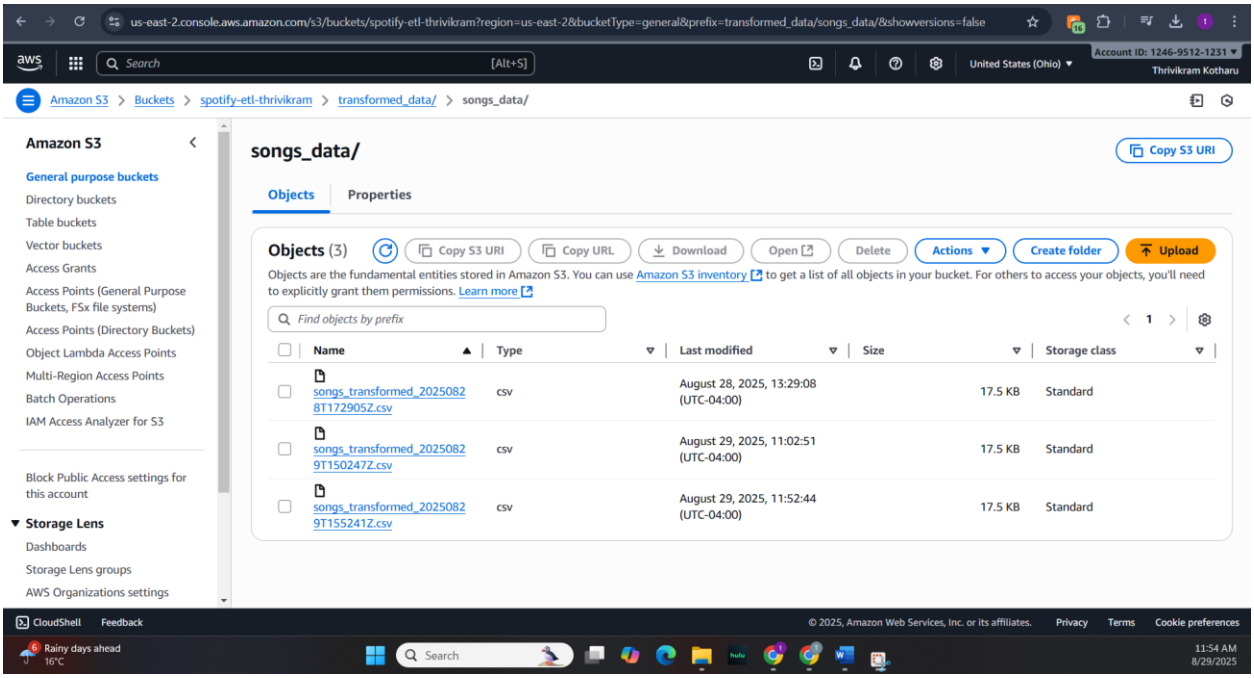


processed data



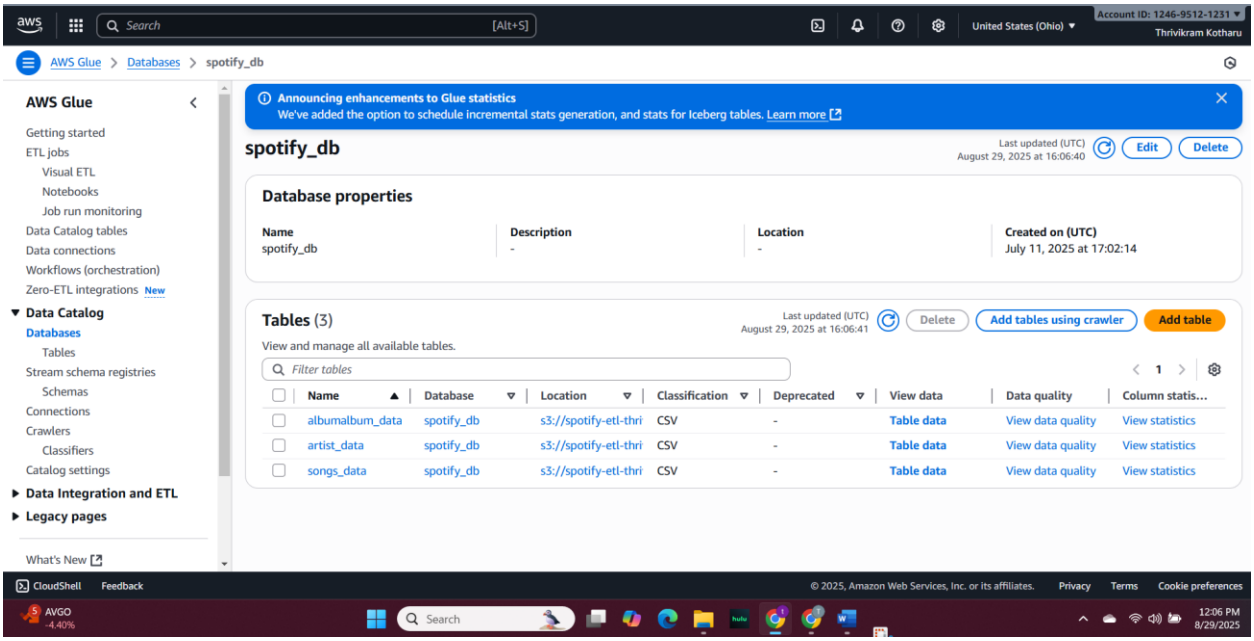



us-east-2.console.aws.amazon.com/s3/buckets/spotify-etl-thrivikram?region=us-east-2&bucketType=general&prefix=transformed_data/artist_data/&showversions=false
Search [Alt+S]
United States (Ohio)
Account ID: 1246-9512-1231
Thrivikram Kotharu
Amazon S3 > Buckets > spotify-etl-thrivikram > transformed_data/ > artist_data/
Amazon S3
General purpose buckets
Directory buckets
Table buckets
Vector buckets
Access Grants
Access Points (General Purpose Buckets, FSx file systems)
Access Points (Directory Buckets)
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
artist_data/
Copy S3 URI
Objects
Properties
Objects (3)
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
artist_transformed_20250828T172905Z.csv
csv
August 28, 2025, 13:29:08 (UTC-04:00)
11.2 KB
Standard
artist_transformed_20250829T150247Z.csv
csv
August 29, 2025, 11:02:51 (UTC-04:00)
11.2 KB
Standard
artist_transformed_20250829T155241Z.csv
csv
August 29, 2025, 11:52:44 (UTC-04:00)
11.2 KB
Standard
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Rainy days ahead
16°C
Search
11:54 AM
8/29/2025

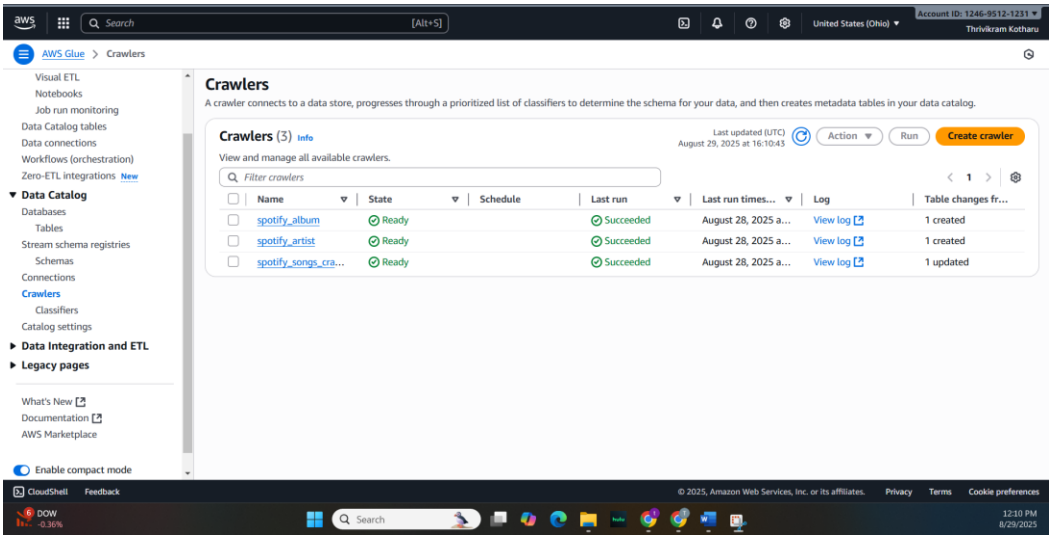

us-east-2.console.aws.amazon.com/s3/buckets/spotify-etl-thrivikram?region=us-east-2&bucketType=general&prefix=transformed_data/songs_data/&showversions=false
Search [Alt+S]
United States (Ohio)
Account ID: 1246-9512-1231
Thrivikram Kotharu
Amazon S3 > Buckets > spotify-etl-thrivikram > transformed_data/ > songs_data/
Amazon S3
General purpose buckets
Directory buckets
Table buckets
Vector buckets
Access Grants
Access Points (General Purpose Buckets, FSx file systems)
Access Points (Directory Buckets)
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
songs_data/
Copy S3 URI
Objects
Properties
Objects (3)
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
songs_transformed_20250828T172905Z.csv
csv
August 28, 2025, 13:29:08 (UTC-04:00)
17.5 KB
Standard
songs_transformed_20250829T150247Z.csv
csv
August 29, 2025, 11:02:51 (UTC-04:00)
17.5 KB
Standard
songs_transformed_20250829T155241Z.csv
csv
August 29, 2025, 11:52:44 (UTC-04:00)
17.5 KB
Standard
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Rainy days ahead
16°C
Search
11:54 AM
8/29/2025

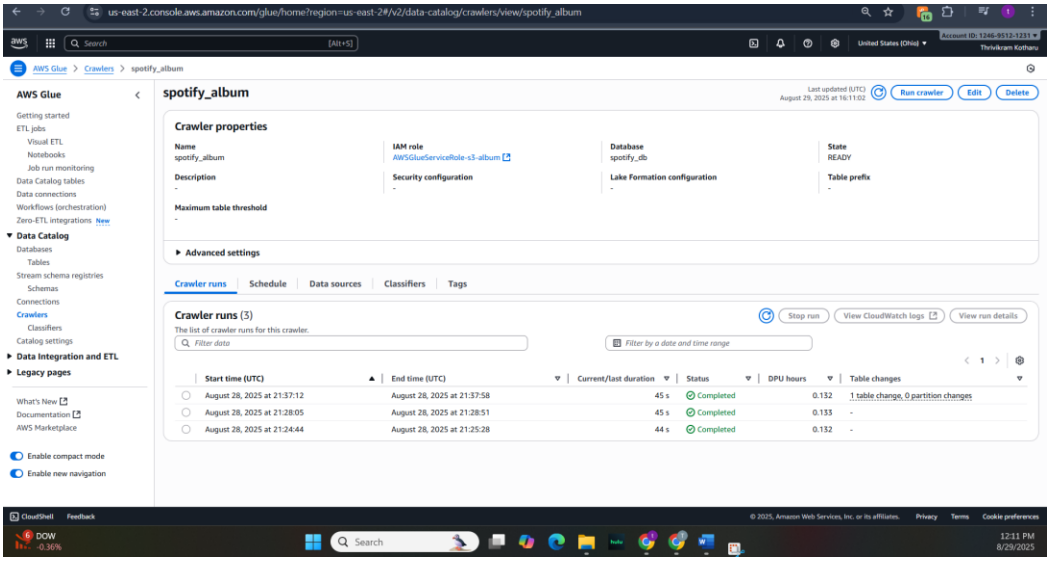
Analysis using AWS Glue, Atena:

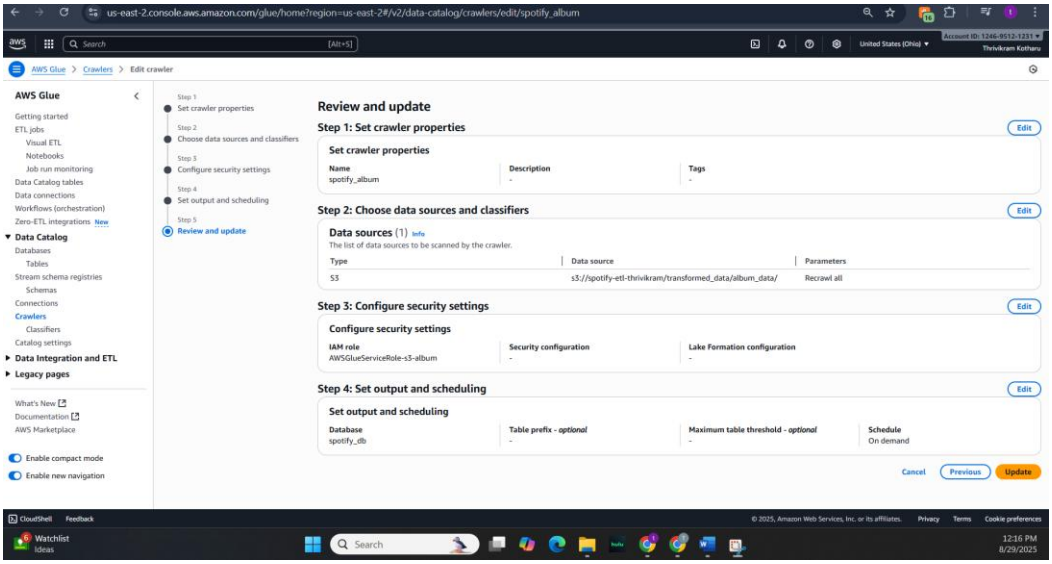
6. After, transformation we got the files ready to perform any analysis required, one of the way we do it is by creating a AWS Glue Database(spotify_db).

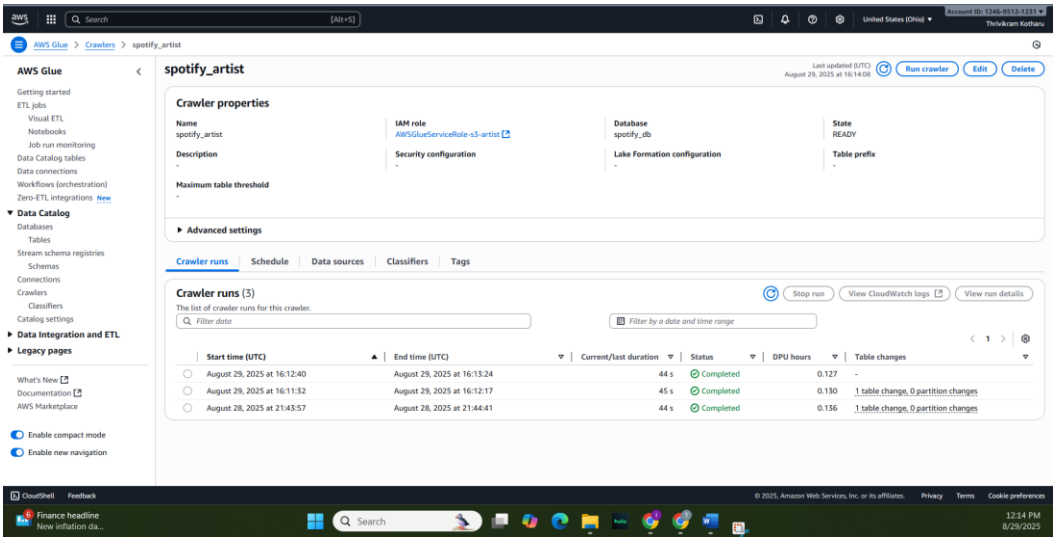


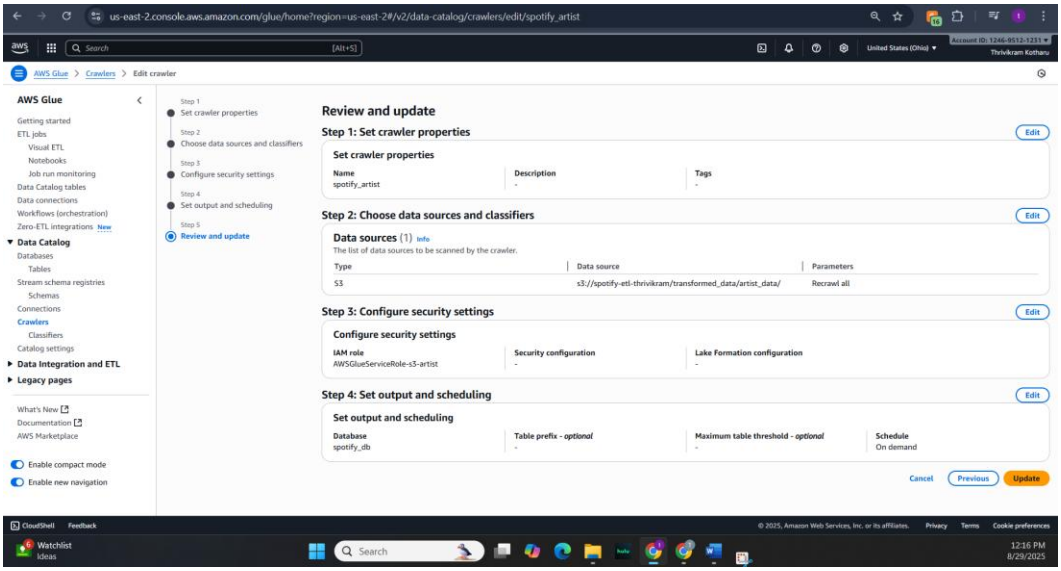
7. Once the database is created, we will use AWS Glue Crawler to extract metadata from the transformed data in the S3. The Crawler needs to be configured with the required IAM role's. Below are the screenshots of Crawler's configuration and I used different crawlers for different folders in S3.

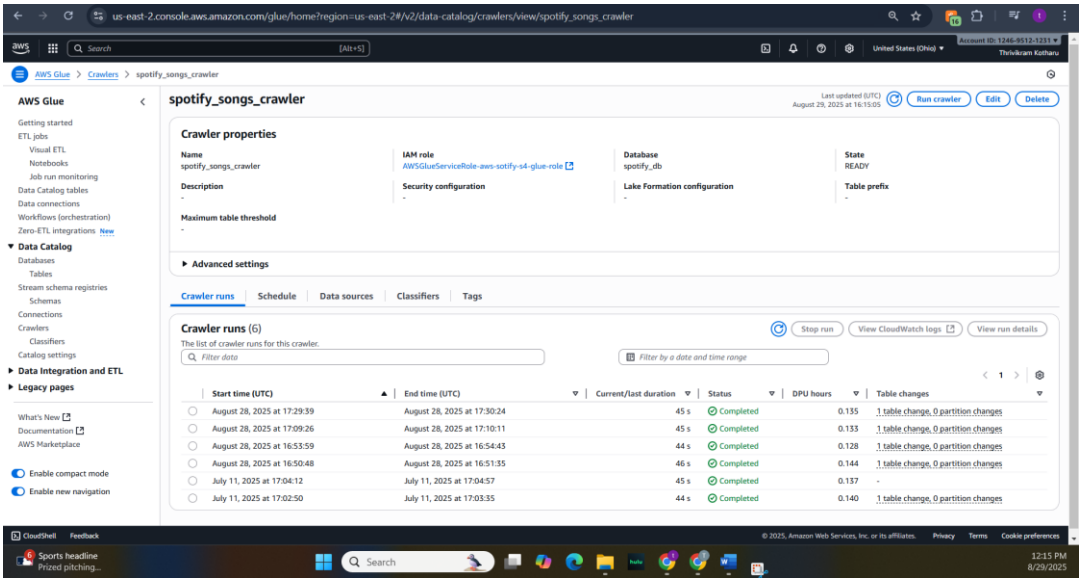


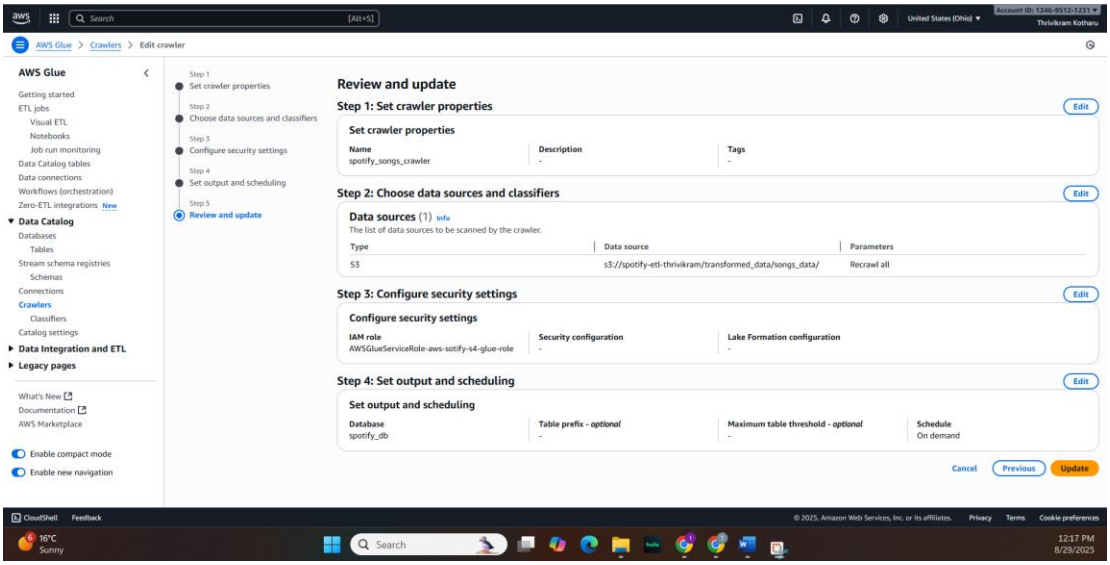

us-east-2.console.aws.amazon.com/glue/home?region=us-east-2#/v2/data-catalog/crawlers/view/spotify_album
AWS Glue > Crawlers > spotify_album
spotify_album
Run crawler
Edit
Delete
Crawler properties
Name
spotify_album
IAM role
AWSGlueServiceRole-s3-album
Database
spotify_db
State
READY
Description
Security configuration
Lake Formation configuration
Table prefix
Maximum table threshold
Advanced settings
Crawler runs
Schedule
Data sources
Classifiers
Tags
Crawler runs (3)
The list of crawler runs for this crawler.
Stop run
View CloudWatch logs
View run details
Start time (UTC)
End time (UTC)
Current/last duration
Status
DPU hours
Table changes
August 28, 2025 at 21:37:12
August 28, 2025 at 21:37:58
45 s
Completed
0.132
1 table change, 0 partition changes
August 28, 2025 at 21:28:05
August 28, 2025 at 21:28:51
45 s
Completed
0.133
August 28, 2025 at 21:24:44
August 28, 2025 at 21:25:28
44 s
Completed
0.132


us-east-2.console.aws.amazon.com/glue/home?region=us-east-2#/v2/data-catalog/crawlers/edit/spotify_album
AWS Glue > Crawlers > Edit crawler
Step 1
Set crawler properties
Step 2
Choose data sources and classifiers
Step 3
Configure security settings
Step 4
Set output and scheduling
Step 5
Review and update
Review and update
Step 1: Set crawler properties
Edit
Set crawler properties
Name
spotify_album
Description
Tags
Step 2: Choose data sources and classifiers
Data sources (1)
Info
The list of data sources to be scanned by the crawler.
Type
Data source
Parameters
S3
s3://spotify-etl-thrivikram/transformed_data/album_data/
Recrawl all
Step 3: Configure security settings
Configure security settings
IAM role
AWSGlueServiceRole-s3-album
Security configuration
Lake Formation configuration
Step 4: Set output and scheduling
Set output and scheduling
Database
spotify_db
Table prefix - optional
Maximum table threshold - optional
Schedule
On demand
Cancel
Previous
Update

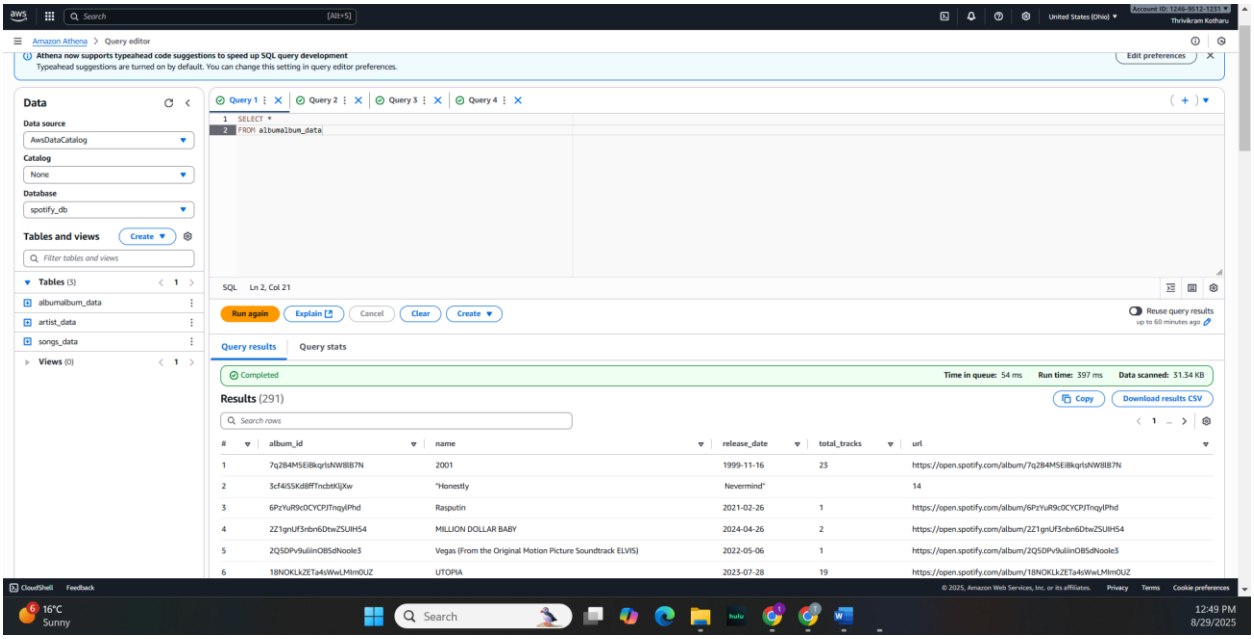

AWS Glue > Crawlers > spotify_artist
spotify_artist
Run crawler
Edit
Delete
Crawler properties
Name
spotify_artist
IAM role
AWSGlueServiceRole-s3-artist
Database
spotify_db
State
READY
Description
Security configuration
Lake Formation configuration
Table prefix
Maximum table threshold
Advanced settings
Crawler runs
Schedule
Data sources
Classifiers
Tags
Crawler runs (3)
The list of crawler runs for this crawler.
Stop run
View CloudWatch logs
View run details
Start time (UTC)
End time (UTC)
Current/last duration
Status
DPU hours
Table changes
August 29, 2025 at 16:12:40
August 29, 2025 at 16:13:24
44 s
Completed
0.127
August 29, 2025 at 16:11:32
August 29, 2025 at 16:12:17
45 s
Completed
0.130
1 table change, 0 partition changes
August 28, 2025 at 21:43:57
August 28, 2025 at 21:44:41
44 s
Completed
0.136
1 table change, 0 partition changes

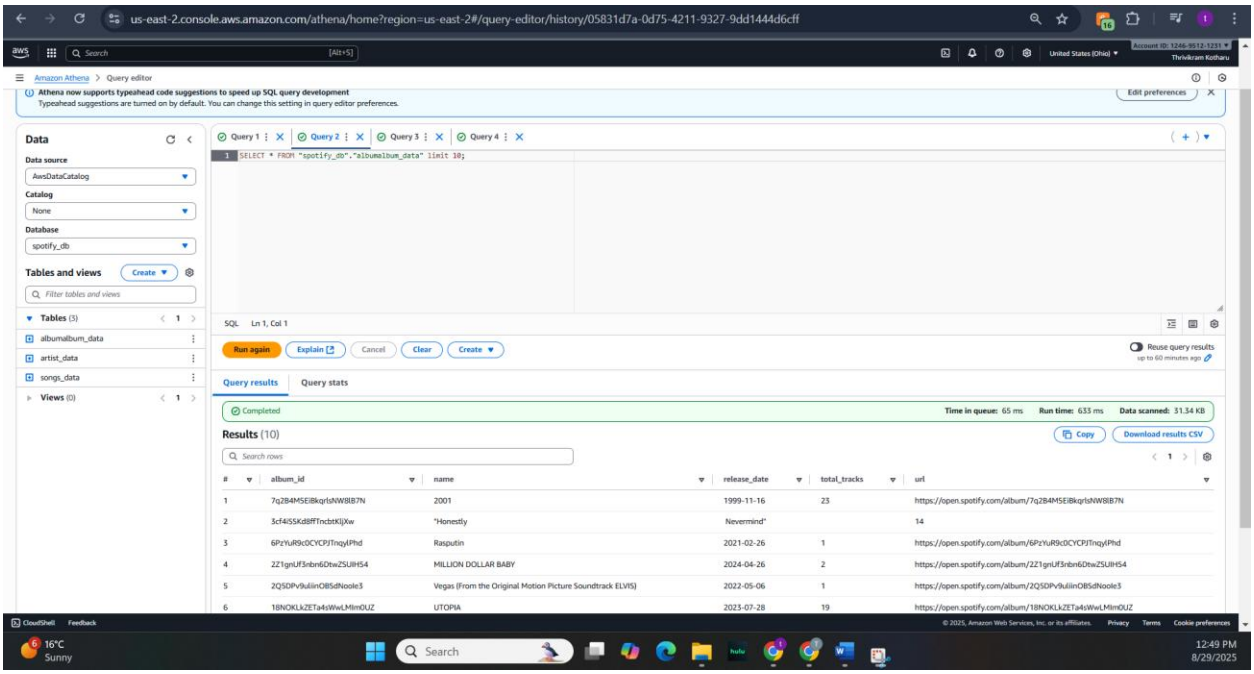

us-east-2.console.aws.amazon.com/glue/home?region=us-east-2#/v2/data-catalog/crawlers/edit/spotify_artist
AWS Glue > Crawlers > Edit crawler
Review and update
Step 1: Set crawler properties
Edit
Set crawler properties
Name
spotify_artist
Description
Tags
Step 2: Choose data sources and classifiers
Edit
Data sources (1) Info
The list of data sources to be scanned by the crawler.
Type
Data source
Parameters
S3
s3://spotify-etl-thrivikram/transformed_data/artist_data/
Recrawl all
Step 3: Configure security settings
Edit
Configure security settings
IAM role
AWSGlueServiceRole-s3-artist
Security configuration
Lake Formation configuration
Step 4: Set output and scheduling
Edit
Set output and scheduling
Database
spotify_db
Table prefix - optional
Maximum table threshold - optional
Schedule
On demand
Cancel
Previous
Update

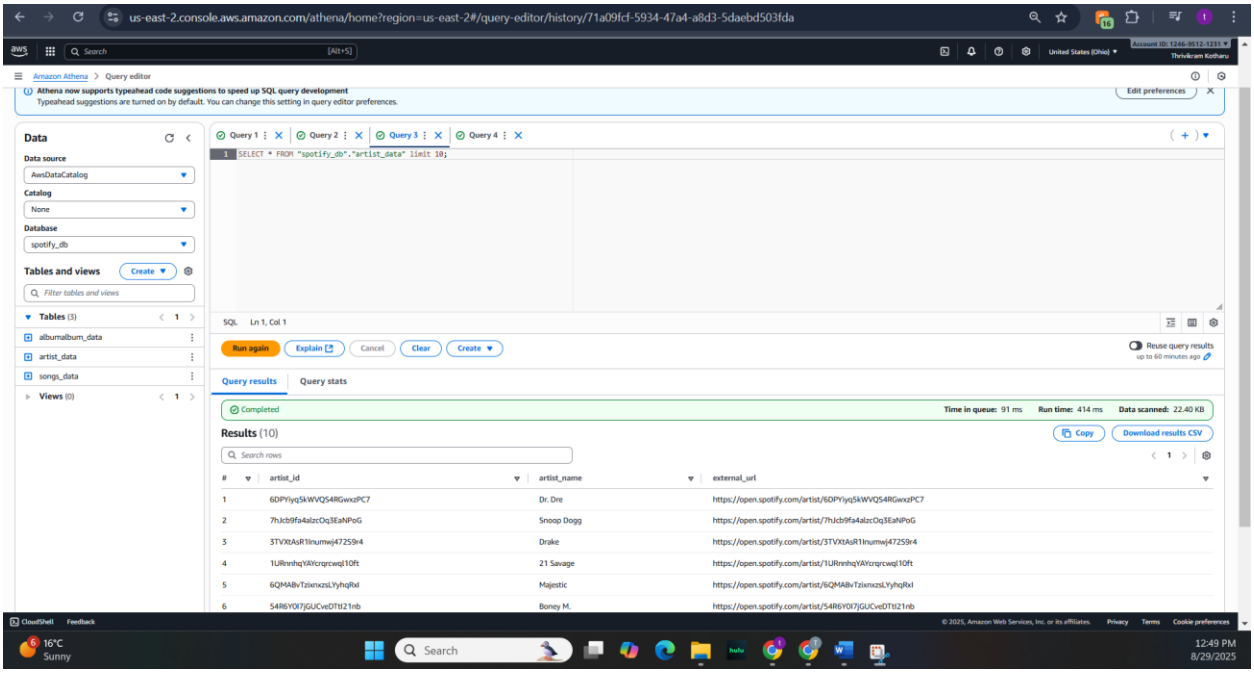

us-east-2.console.aws.amazon.com/glue/home?region=us-east-2#/v2/data-catalog/crawlers/view/spotify_songs_crawler
AWS Glue > Crawlers > spotify_songs_crawler
spotify_songs_crawler
Run crawler
Edit
Delete
Crawler properties
Name
spotify_songs_crawler
IAM role
AWSGlueServiceRole-aws-sotify-s4-glue-role
Database
spotify_db
State
READY
Description
Security configuration
Lake Formation configuration
Table prefix
Maximum table threshold
Advanced settings
Crawler runs
Schedule
Data sources
Classifiers
Tags
Crawler runs (6)
The list of crawler runs for this crawler.
Stop run
View CloudWatch logs
View run details
Start time (UTC)
End time (UTC)
Current/last duration
Status
DPU hours
Table changes
August 28, 2025 at 17:29:39
August 28, 2025 at 17:30:24
45 s
Completed
0.135
1 table change, 0 partition changes
August 28, 2025 at 17:09:26
August 28, 2025 at 17:10:11
45 s
Completed
0.133
1 table change, 0 partition changes
August 28, 2025 at 16:53:59
August 28, 2025 at 16:54:43
44 s
Completed
0.128
1 table change, 0 partition changes
August 28, 2025 at 16:50:48
August 28, 2025 at 16:51:35
46 s
Completed
0.144
1 table change, 0 partition changes
July 11, 2025 at 17:04:12
July 11, 2025 at 17:04:57
45 s
Completed
0.137
July 11, 2025 at 17:02:50
July 11, 2025 at 17:03:35
44 s
Completed
0.140
1 table change, 0 partition changes

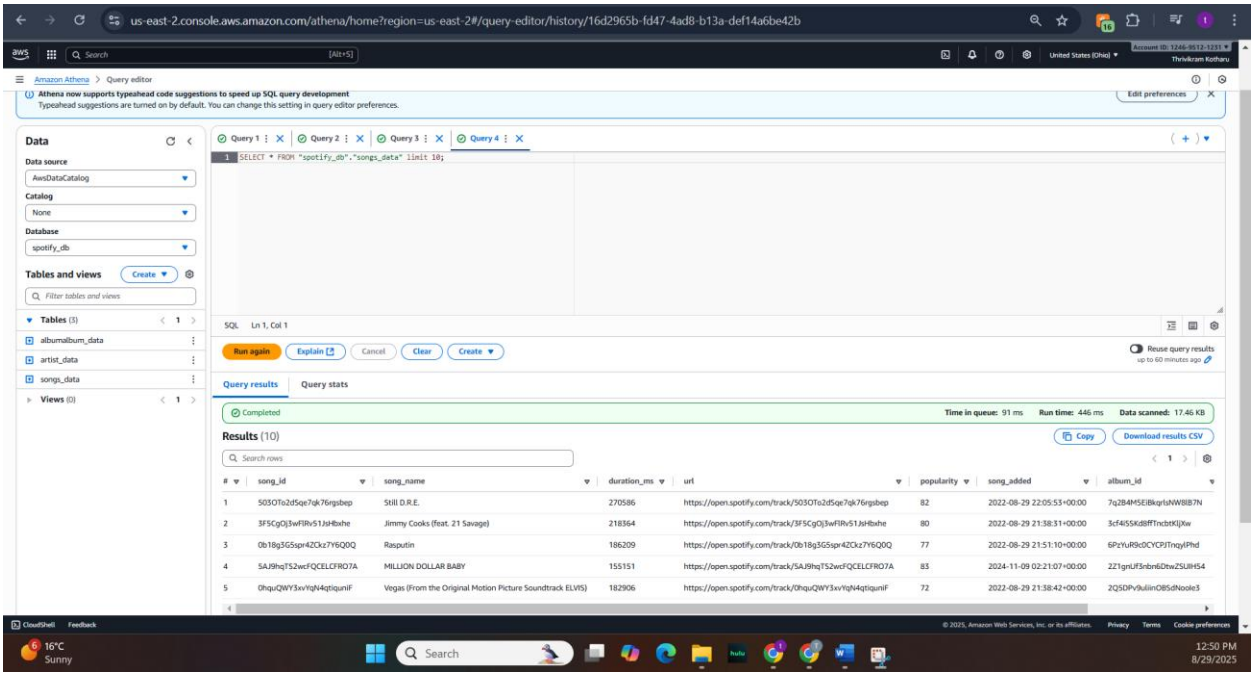

AWS Glue > Crawlers > Edit crawler
Review and update
Step 1: Set crawler properties
Edit
Set crawler properties
Name
spotify_songs_crawler
Description
Tags
Step 2: Choose data sources and classifiers
Edit
Data sources (1) Info
The list of data sources to be scanned by the crawler.
Type
Data source
Parameters
S3
s3://spotify-etl-thrivikram/transformed_data/songs_data/
Recrawl all
Step 3: Configure security settings
Edit
Configure security settings
IAM role
AWSGlueServiceRole-aws-sotify-s4-glue-role
Security configuration
Lake Formation configuration
Step 4: Set output and scheduling
Edit
Set output and scheduling
Database
spotify_db
Table prefix - optional
Maximum table threshold - optional
Schedule
On demand
Cancel
Previous
Update

8. Now, that the tables are added to the database we created in the AWS glue, we can now perform different Query's to perform different analytics on this data. Below is the screenshot of the AWS Athena in action performing various data analysis.









Conclusion

This ETL pipeline successfully demonstrates how Spotify playlist data can be ingested, transformed, and analyzed in a fully automated, serverless environment.

Key achievements

- **Automated weekly data refresh** with CloudWatch triggers.
- **Scalable transformations** developed in Jupyter and deployed in Lambda.
- **Schema discovery and analytics** via Glue Data Catalog and Athena.

Future Plans

- In the next phase, this pipeline will be extended to integrate with **Snowflake**.
- Transformed S3 data will be ingested into Snowflake for **Data Warehousing**.
- Snowflake's **OLAP capabilities** will support advanced analytics, faster queries, and complex aggregations.
- This will provide a more robust foundation for **business intelligence dashboards** and reporting tools.