



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.

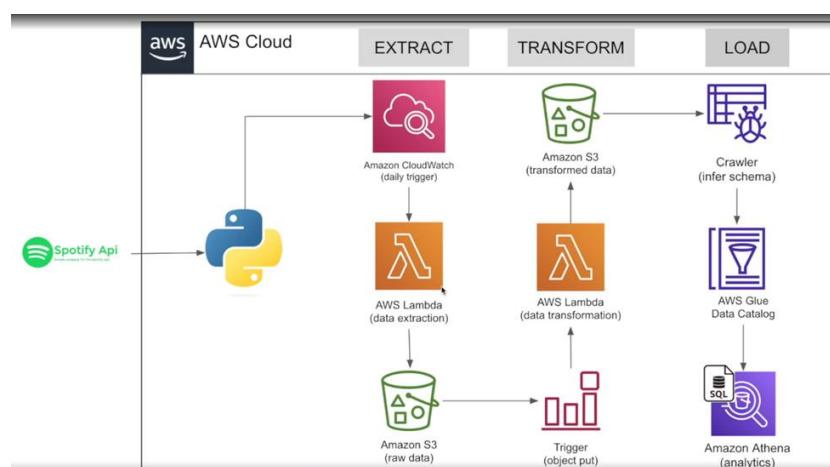
The goal of this project is to build a fully automated ETL pipeline for Spotify playlist data using AWS services. The pipeline is designed to:

- 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.

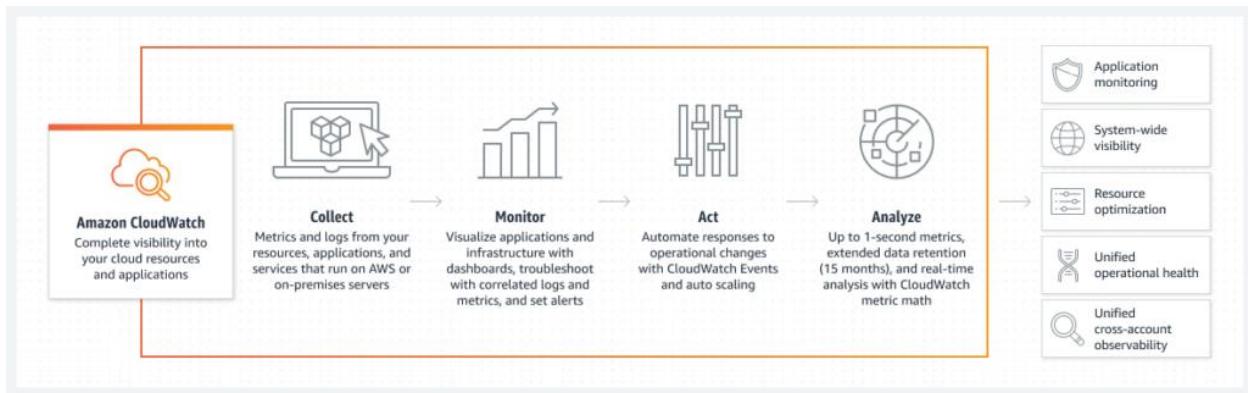
Architecture & Workflow

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, event-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

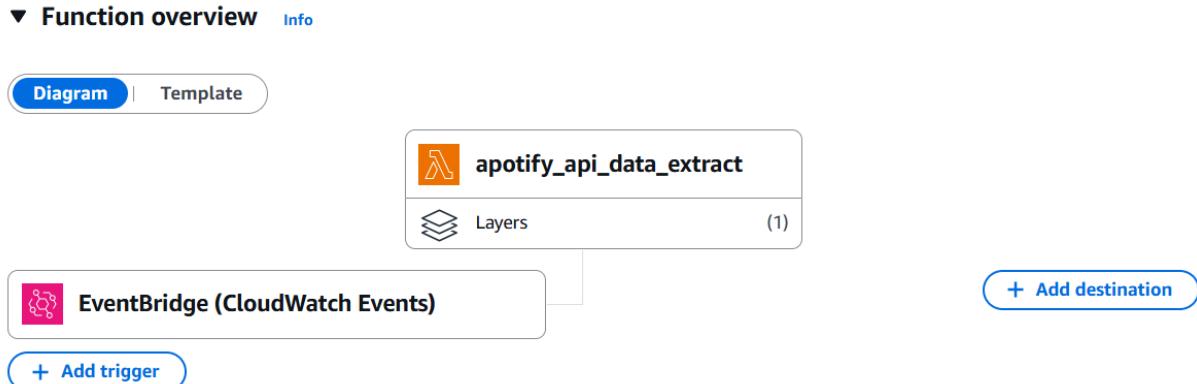
Data Extraction:

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

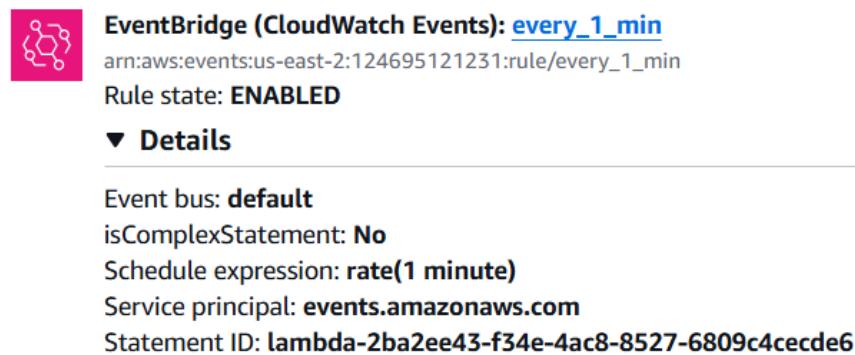
Code:

```
lambda_function.py X
❶ lambda_function.py
❷
❸ import json
❹ import os
❺ import spotipy
❻ from spotipy.oauth2 import SpotifyClientCredentials
❼ import boto3
❼ from datetime import datetime
❼
❼ def lambda_handler(event, context):
❼
❼     client_id = os.environ.get('client_id')
❼     client_secret = os.environ.get('client_secret')
❼
❼     client_credentials_manager = SpotifyClientCredentials(client_id=client_id, client_secret=client_secret)
❼     sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager)
❼     playlists = sp.user_playlists('spotify')
❼
❼     playlist_link = "https://open.spotify.com/playlist/2SM6rnizl84fEyNCB5KMQB"
❼     playlist_URI = playlist_link.split("/")[-1].split("?")[-1]
❼
❼     spotify_data = sp.playlist_tracks(playlist_URI)
❼
❼     client = boto3.client('s3')
❼
❼     filename = "spotify_raw_" + str(datetime.now()) + ".json"
❼
❼     client.put_object(
❼         Bucket="spotify-etl-thrivikram",
❼         Key="raw_data/to_processed/" + filename,
❼         Body=json.dumps(spotify_data)
❼     )
```

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.

The screenshot shows the AWS Lambda console interface. The top navigation bar includes links for AWS Lambda, Functions, transformation, Throttle, Copy ARN, Actions, Export to Infrastructure Composer, and Download. The main area displays the 'transformation' function overview, showing a diagram with an S3 source and a Lambda target. The 'Configuration' tab is active, revealing a 'Triggers' section. One trigger is listed: 'S3: spotify-eti-thrivikram' (arn:aws:s3:::raw_data/to_processed). The trigger details show the event type as 's3:ObjectCreated:*', the notification name as '96434ff-886c-4563-805f-6273e287089', and the prefix as 'raw_data/to_processed/'. The trigger was last modified 23 hours ago.

to_processed data

The screenshot shows the Amazon S3 console with the path 'Amazon S3 > Buckets > spotify-eti-thrivikram > raw_data /> to_processed/'. The left sidebar includes options like 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, Storage Lens, Dashboards, Storage Lens groups, and AWS Organizations settings. The main area displays the 'to_processed/' folder, which contains three objects: '02:28:38.061845.json', '14:52:38.195934.json', and '14:55:34.112739.json'. All three files were uploaded on August 29, 2025, at different times between 14:52:38 and 14:55:34 UTC-04:00. The file sizes are 434.5 KB each, and they are stored in the Standard storage class.

processed data

Amazon S3 < transformed_data/

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](#)

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	album_data/	Folder	-	-	-
<input type="checkbox"/>	artist_data/	Folder	-	-	-
<input type="checkbox"/>	songs_data/	Folder	-	-	-

Block Public Access settings for this account

▼ Storage Lens

- Dashboards
- Storage Lens groups
- AWS Organizations settings

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 11:01 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/album_data/8showversions=false

aws Search [Alt+S] Account ID: 1246-9512-1231 United States (Ohio) Thrivikram Kothur

Amazon S3 > Buckets > spotify-etl-thrivikram > transformed_data/ > album_data/

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](#)

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	album_transformed_2025082 8T172905Z.csv	csv	August 28, 2025, 13:29:08 (UTC-04:00)	10.4 KB	Standard
<input type="checkbox"/>	album_transformed_2025082 9T150247Z.csv	csv	August 29, 2025, 11:02:51 (UTC-04:00)	10.4 KB	Standard
<input type="checkbox"/>	album_transformed_2025082 9T155241Z.csv	csv	August 29, 2025, 11:52:44 (UTC-04:00)	10.4 KB	Standard

Block Public Access settings for this account

▼ Storage Lens

- Dashboards
- Storage Lens groups
- AWS Organizations settings

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 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/artist_data/&showversions=false

Amazon S3 Buckets spotify-etl-thrivikram transformed_data/ artist_data/

Objects (3) Actions Create folder Upload

Name	Type	Last modified	Size	Storage class
artist_transformed_20250828_T172905Z.csv	csv	August 28, 2025, 13:29:08 (UTC-04:00)	11.2 KB	Standard
artist_transformed_20250829_T150247Z.csv	csv	August 29, 2025, 11:02:51 (UTC-04:00)	11.2 KB	Standard
artist_transformed_20250829_T155241Z.csv	csv	August 29, 2025, 11:52:44 (UTC-04:00)	11.2 KB	Standard

Find objects by prefix

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 11:54 AM 8/29/2025

Rainy days ahead 16°C

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

Amazon S3 Buckets spotify-etl-thrivikram transformed_data/ songs_data/

Objects (3) Actions Create folder Upload

Name	Type	Last modified	Size	Storage class
songs_transformed_20250828_8T172905Z.csv	csv	August 28, 2025, 13:29:08 (UTC-04:00)	17.5 KB	Standard
songs_transformed_20250829_9T150247Z.csv	csv	August 29, 2025, 11:02:51 (UTC-04:00)	17.5 KB	Standard
songs_transformed_20250829_9T155241Z.csv	csv	August 29, 2025, 11:52:44 (UTC-04:00)	17.5 KB	Standard

Find objects by prefix

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 11:54 AM 8/29/2025

Rainy days ahead 16°C

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).

The screenshot shows the AWS Glue console interface. On the left, there's a navigation sidebar with options like 'Getting started', 'ETL jobs', 'Visual ETL', 'Notebooks', 'Job run monitoring', 'Data Catalog tables', 'Data connections', 'Workflows (orchestration)', and 'Zero-ETL integrations'. Under 'Data Catalog', there are sections for 'Databases', 'Tables', 'Stream schema registries', 'Schemas', 'Connections', 'Crawlers', 'Classifiers', and 'Catalog settings'. Below these, there are sections for 'Data Integration and ETL' and 'Legacy pages'. At the bottom of the sidebar, there's a 'What's New' section and links for 'CloudShell' and 'Feedback'. The main content area is titled 'spotify_db' and shows 'Database properties' with fields for 'Name' (spotify_db), 'Description' (empty), 'Location' (empty), and 'Created on (UTC)' (July 11, 2025 at 17:02:14). Below this is a 'Tables (3)' section with a table showing three entries:

Name	Database	Location	Classification	Deprecated	Action	Data quality	Column stats...
album_data	spotify_db	s3://spotify-etl-thri	CSV	-	Table data	View data quality	View statistics
artist_data	spotify_db	s3://spotify-etl-thri	CSV	-	Table data	View data quality	View statistics
songs_data	spotify_db	s3://spotify-etl-thri	CSV	-	Table data	View data quality	View statistics

At the bottom of the page, there are links for 'CloudShell', 'Feedback', and a copyright notice: '© 2025, Amazon Web Services, Inc. or its affiliates.' followed by 'Privacy', 'Terms', and 'Cookie preferences'. The status bar at the bottom right shows '12:06 PM' and '8/29/2025'.

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.

The screenshot shows the AWS Glue console interface, similar to the previous one but with a different focus. The left sidebar has the same structure, including the 'Crawlers' section under 'Data Catalog'. The main content area is titled 'Crawlers' and shows a table with three entries:

Name	State	Schedule	Last run	Last run times...	Action	Log	Table changes fr...
spotify_album	Ready		2025-08-28T16:10:45Z	2025-08-28T16:10:45Z	View log		1 created
spotify_artist	Ready		2025-08-28T16:10:45Z	2025-08-28T16:10:45Z	View log		1 created
spotify_songs_cra...	Ready		2025-08-28T16:10:45Z	2025-08-28T16:10:45Z	View log		1 updated

At the bottom of the page, there are links for 'CloudShell', 'Feedback', and a copyright notice: '© 2025, Amazon Web Services, Inc. or its affiliates.' followed by 'Privacy', 'Terms', and 'Cookie preferences'. The status bar at the bottom right shows '12:10 PM' and '8/29/2025'.

The screenshot shows the AWS Glue Crawler configuration page for the crawler 'spotify_album'. The left sidebar includes links for AWS Glue, Crawlers, Data Catalog, Data Integration and ETL, and Legacy pages. The main content area displays the 'Crawler properties' section with tabs for IAM role, Security configuration, Database, State, and Table prefix. It also shows the 'Maximum table threshold' and 'Advanced settings' sections. Below this, the 'Crawler runs' tab is selected, showing three completed runs on August 28, 2025, at various times. A 'Filter by a date and time range' input field is present. The bottom of the page includes CloudShell, Feedback, and standard footer links.

Last updated (UTC)
August 29, 2025 at 16:11:02

Run crawler Edit Delete

Crawler properties

Name: spotify_album

IAM role: AWSGlueServiceRole-s3-album

Description:

Security configuration:

Database: spotify_db

State: READY

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:

Filter by a date and time range

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	-

CloudShell Feedback

Search

12:31 PM
8/29/2025

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

Review and update

Step 1: Set crawler properties

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 S3	Data source s3://spotify-etl-thivikram/transformed_data/album_data/	Parameters 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**

cloudshell Feedback

Watchlist Ideas

Search

16 PM 8/20/2025

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

AWS Glue > Crawlers > spotify_artist

AWS Glue

Getting started
ETL jobs
Visual ETL
Notebooks
Job run monitoring
Data catalog tables
Data connections
Workflows (orchestration)
Zero-ETL Integrations [New](#)

Crawler properties

Name: spotify_artist
Job run monitoring: Enabled
Database: spotify_db
Lake Formation configuration: None
State: READY
Table prefix: None

Maximum table threshold: 1000

Advanced settings

Crawler runs (3)

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

Filter data Filter by a date and time range

Stop run View CloudWatch logs View run details

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Finance headline New inflation da...

Last updated (UTC) August 29, 2025 at 16:14:08 [Run crawler](#) [Edit](#) [Delete](#)

Account ID: 1234-9512-1231 Thirukaran Kathar...

https://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

Set crawler properties

Name	spotify_artist	Description	Tags
------	----------------	-------------	------

Step 2: Choose data sources and classifiers

Data sources (1) info

Type	Data source	Parameters
S3	s3://spotify-etl-thrivikram/transformed_data/artist_data/	Recrawl all

Step 3: Configure security settings

Configure security settings

IAM role	AWSGlueServiceRole-s3-artist	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**

CloudShell Feedback Search United States (Ohio) Account ID: 1234-9876-5432-1234 Thirvikram Katharu 12:16 PM 8/29/2025

https://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

Crawler properties

Name	spotify_songs_crawler	IAM role	AWSGlueServiceRole-aws-spotify-s4-glue-role	Database	spotify_db	State	
Description		Security configuration		Lake Formation configuration		READY	Table prefix
Maximum table threshold	-						

Advanced settings

Crawler runs (6)

The list of crawler runs for this crawler.

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	0.135	1 table change, 0 partition changes
August 28, 2025 at 16:53:59	August 28, 2025 at 16:54:45	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

Run crawler **Edit** **Delete**

CloudShell Feedback Search United States (Ohio) Account ID: 1234-9876-5432-1234 Thirvikram Katharu 12:15 PM 8/29/2025

https://us-east-2.console.aws.amazon.com/glue/home?region=us-east-2#/v2/data-catalog/crawlers/edit/spotify_songs_crawler

AWS Glue > Crawlers > Edit crawler

Review and update

Step 1: Set crawler properties

Set crawler properties

Name	spotify_songs_crawler	Description	Tags
------	-----------------------	-------------	------

Step 2: Choose data sources and classifiers

Data sources (1) info

Type	Data source	Parameters
S3	s3://spotify-etl-thrivikram/transformed_data/songs_data/	Recrawl all

Step 3: Configure security settings

Configure security settings

IAM role	AWSGlueServiceRole-aws-spotify-s4-glue-role	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**

CloudShell Feedback Search United States (Ohio) Account ID: 1234-9876-5432-1234 Thirvikram Katharu 12:17 PM 8/29/2025

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.

The screenshot shows the AWS Athena Query Editor interface. The left sidebar displays the 'Data' section with 'Data source' set to 'AwsDataCatalog', 'Catalog' to 'None', and 'Database' to 'spotify_db'. Under 'Tables and views', there are three tables: 'albumalbum_data', 'artist_data', and 'songs_data'. The 'Views (0)' section is empty. The main area shows four tabs: 'Query 1', 'Query 2', 'Query 3', and 'Query 4'. The 'Query 1' tab contains the following SQL code:

```
1 SELECT *
2 FROM albumalbum_data
```

The 'Results' tab shows the output of the query:

#	album_id	name	release_date	total_tracks	url
1	7q2B4M5EiBkqrtsNWBB7N	2001	1999-11-16	23	https://open.spotify.com/album/7q2B4M5EiBkqrtsNWBB7N
2	3c74f5SkdbffTndtKjXw	'Honesty'	Nevermind	14	
3	6Pz1uR9kOcYCpJnqyPhd	Rasputin	2021-02-26	1	https://open.spotify.com/album/6Pz1uR9kOcYCpJnqyPhd
4	221gnUf3nbr6DtwZSfH54	MILLION DOLLAR BABY	2024-04-26	2	https://open.spotify.com/album/221gnUf3nbr6DtwZSfH54
5	2Q5DPv9ulinOB5nole5	Vegas (From the Original Motion Picture Soundtrack ELVIS)	2022-05-06	1	https://open.spotify.com/album/2Q5DPv9ulinOB5nole5
6	1BNOKLj2ETa4sWwLMmDfZ	UTOPIA	2023-07-28	19	https://open.spotify.com/album/1BNOKLj2ETa4sWwLMmDfZ

At the bottom right of the results table, it says 'Time in queue: 54 ms Run time: 397 ms Data scanned: 51.34 KB'. There are 'Copy' and 'Download results CSV' buttons.

This screenshot shows the same AWS Athena Query Editor interface as the previous one, but with a different query. The 'Query 1' tab contains the following SQL code:

```
1 SELECT * FROM "spotify_db"."albumalbum_data" limit 10;
```

The 'Results' tab shows the output of the query, which is identical to the one above, displaying 10 rows of album data. The results table includes the same columns and data as the first screenshot.

The screenshot shows the AWS Athena Query Editor interface. On the left, the Data source is set to 'AwsDataCatalog'. Under 'Tables and views', there are three tables listed: 'album_album_data', 'artist_data', and 'songs_data'. The 'artist_data' table is selected. The main area displays a query result for 'artist_data' with a limit of 10 rows. The results show columns: #, artist_id, artist_name, and external_url. The results are as follows:

#	artist_id	artist_name	external_url
1	6DPHyq5kWVQ54RGenzPC7	Dr. Dre	https://open.spotify.com/artist/6DPHyq5kWVQ54RGenzPC7
2	7hJcbfFaSalzOqfEaPnG	Snoop Dogg	https://open.spotify.com/artist/7hJcbfFaSalzOqfEaPnG
3	3TVX2AsR1numyj47259h4	Drake	https://open.spotify.com/artist/3TVX2AsR1numyj47259h4
4	1URmhpqfWrcqeq10ft	21 Savage	https://open.spotify.com/artist/1URmhpqfWrcqeq10ft
5	6GMA4Bv7SmazslYjhqRd	Majestic	https://open.spotify.com/artist/6GMA4Bv7SmazslYjhqRd
6	54R6Y0f7GUUcveDT121rb	Boney M.	https://open.spotify.com/artist/54R6Y0f7GUUcveDT121rb

This screenshot shows the same AWS Athena Query Editor interface, but the 'songs_data' table is now selected. The query result for 'songs_data' with a limit of 10 rows is displayed. The results show columns: #, song_id, song_name, duration_ms, url, popularity, song_added, and album_id. The results are as follows:

#	song_id	song_name	duration_ms	url	popularity	song_added	album_id
1	503To2z5q7jk76rgsbeP	Still D.R.E.	27056	https://open.spotify.com/track/503To2z5q7jk76rgsbeP	82	2022-08-29 22:05:53+00:00	7q2B4H5E8kqrsNW87N
2	3F5Cg0jSwfIRv5tJhBhze	Jimmy Cooks (feat. 21 Savage)	218364	https://open.spotify.com/track/3F5Cg0jSwfIRv5tJhBhze	80	2022-08-29 21:58:31+00:00	3cf455ckdPfTnctkXw
3	Ob18g3G5spr4ZCkz7y6Q0Q	Rasputin	186209	https://open.spotify.com/track/Ob18g3G5spr4ZCkz7y6Q0Q	77	2022-08-29 21:51:10+00:00	6PzYuRsk0C0CYpTrnqyPhd
4	5AJ9hqf52weFQCELFRO7A	MILLION DOLLAR BABY	155151	https://open.spotify.com/track/5AJ9hqf52weFQCELFRO7A	83	2024-11-09 02:21:07+00:00	271grnfJfnbm6Ctw2LHs4
5	0QuQWY3xVqNq4tqunIF	Vegas (From the Original Motion Picture Soundtrack ELVIS)	182906	https://open.spotify.com/track/0QuQWY3xVqNq4tqunIF	72	2022-08-29 21:58:42+00:00	2Q5DpVqUlnOB8dhk0k3

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.