Cloud Data Pipeline Mastery: End-to-End Analytics with AWS

"Building Scalable and Versatile Data Solutions for Any Dataset"

Sai Srivatsa Thangallapelly

Abstract

This project focuses on building an end-to-end data engineering pipeline using AWS services to analyze and visualize Spotify data. The project demonstrates how to ingest, process, store, and visualize large datasets, making them suitable for real-world applications in data-driven decision-making. The use of AWS Glue, S3, Athena, and QuickSight allows for an efficient, scalable, and cost-effective solution to process and analyze data, offering insights that can benefit various stakeholders in the music industry. This project was inspired and guided by the tutorials on the Date with Data.

Introduction

In this project, we will build an end-to-end data engineering pipeline using AWS cloud services. The project will focus on processing and analyzing Spotify data using various AWS tools like S3, Glue, Athena, and QuickSight.

Project Architecture Overview

1. **Staging Layer**: Raw data is stored in an S3 bucket.
2. **ETL Pipeline**: AWS Glue processes and transfers data from the staging layer to the data warehouse.
3. **Data Warehouse**: Processed data is stored in another S3 bucket.
4. **Data Catalog**: AWS Glue Crawler creates a database and tables for the data warehouse.
5. **Data Analysis**: AWS Athena queries the processed data.
6. **Data Visualization**: AWS QuickSight visualizes the data.

Prerequisites

- An AWS account
- Basic understanding of AWS services like S3, Glue, Athena, and QuickSight

AWS Services Used

- **Amazon S3**: For storing raw and processed data.
- **AWS Glue**: For building and managing ETL pipelines.
- **AWS Athena**: For querying data using SQL-like syntax.
- **AWS QuickSight**: For visualizing data.

Data Source

The data used in this project is sourced from the Spotify Dataset 2023 available on Kaggle. The dataset, created by Tony Gordon Jr., includes detailed information about Spotify albums, artists, tracks, and various audio features like danceability, energy, loudness, and more. The dataset is available in CSV format and has been pre-processed for use in this project.

Data Description

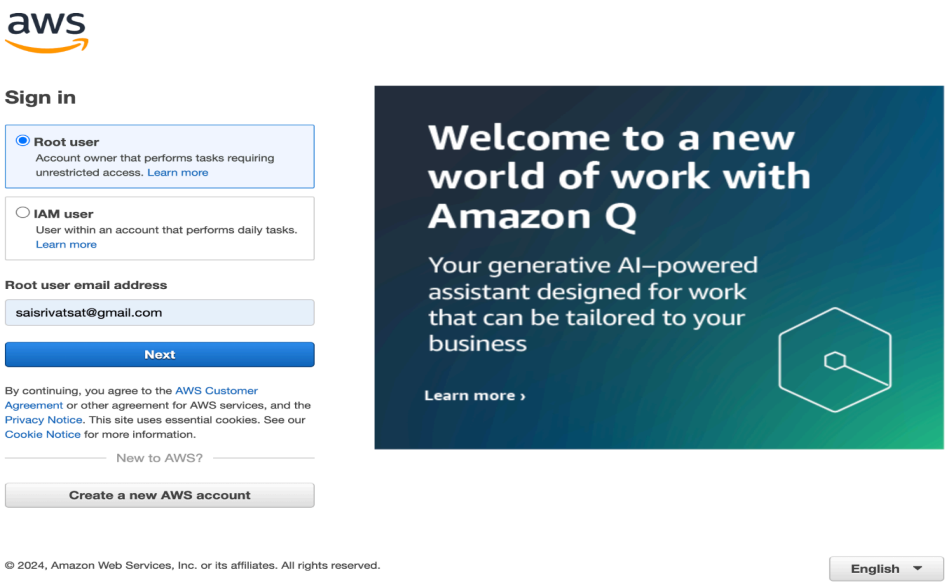
- **Albums**: Contains details of all the albums, including album ID, name, popularity, and release date.
- **Artists**: Contains information about the artists, including their names, number of followers, and genres.
- **Tracks**: Contains track-level data, including track ID, popularity, and other features like danceability and energy.
- **Spotify Features**: Contains various audio features like loudness, mode, speechiness, and valence.

Hands-On Steps

Step 1: Setting Up AWS IAM User

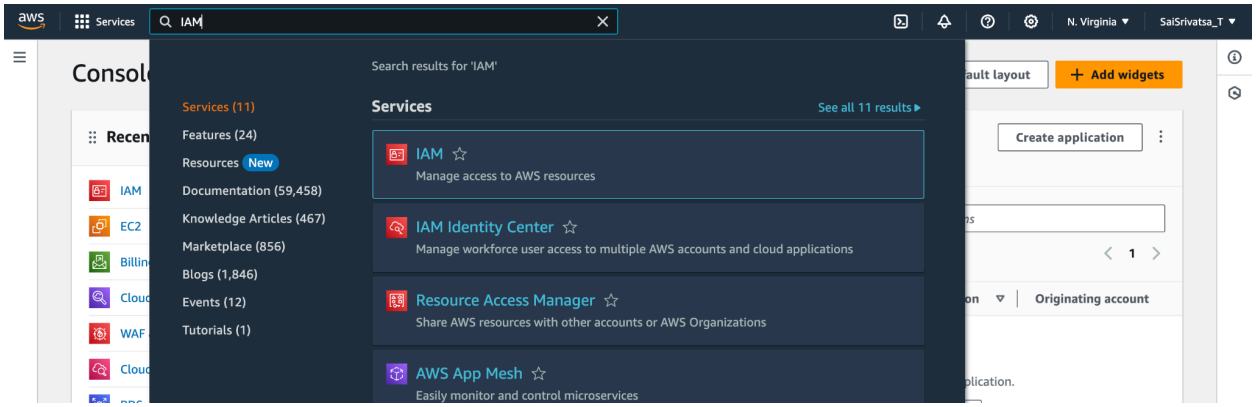
1. **Log in to AWS Management Console:**

- Open your web browser and go to the AWS Management Console.
- Log in using your root account credentials.

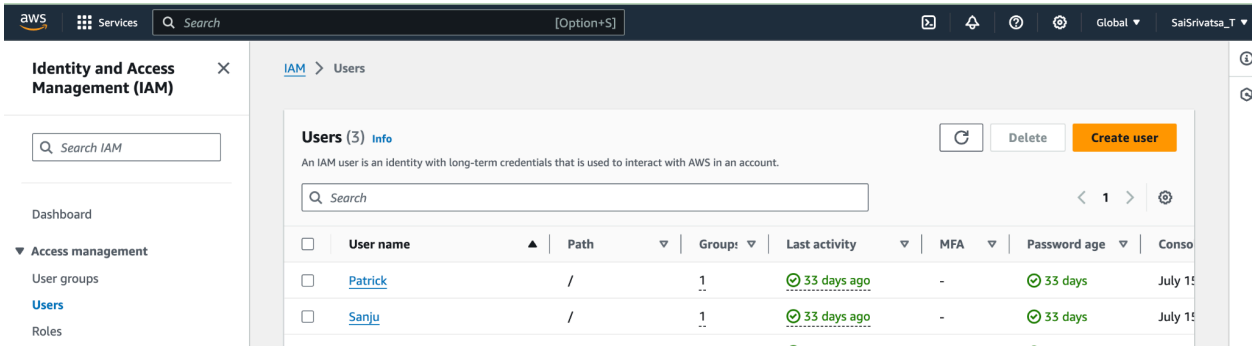


2. **Create a New IAM User:**

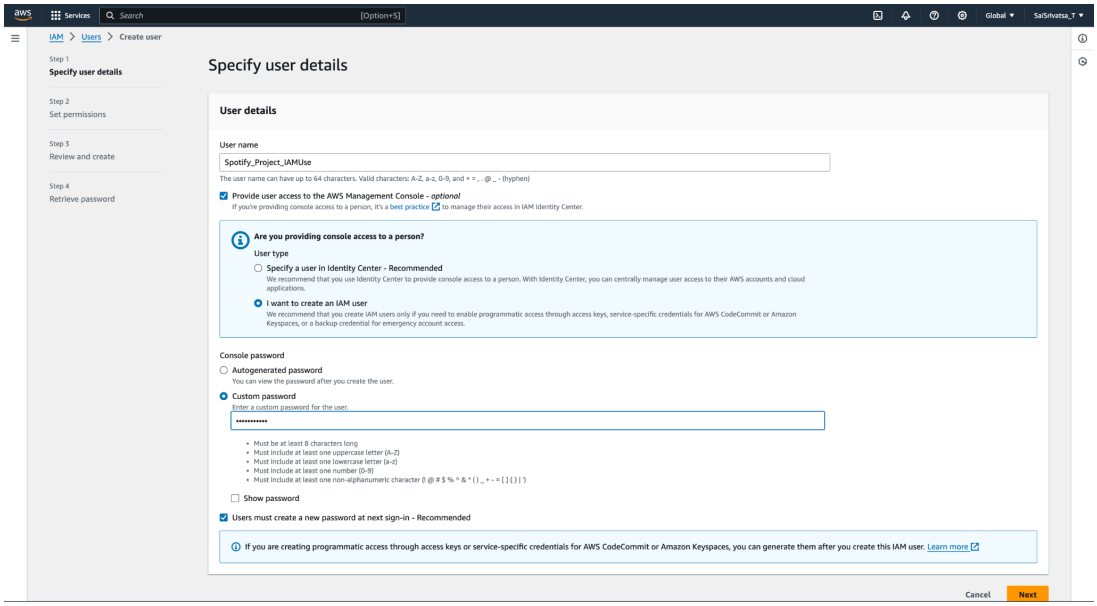
- In the AWS Management Console, search for “IAM” in the search bar and select the IAM service.



- In the IAM dashboard, click on "Users" from the left-hand menu and then click "Create user."

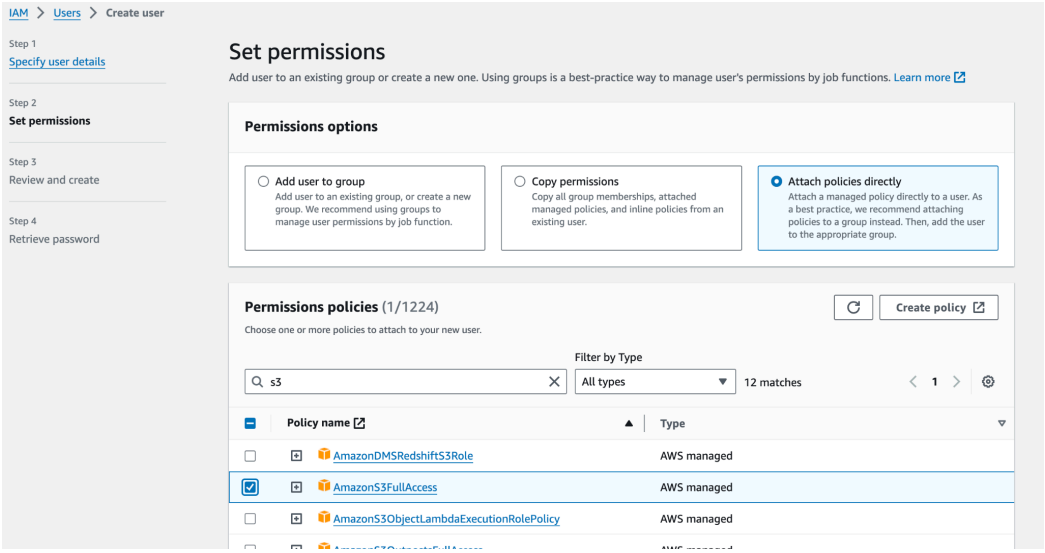


- **Enter the Following Details**
  - **Username**: Enter project_user.
  - **Access Type**: Select "Provide user access to the AWS Management Console - optional"
  - **Console Password**: Choose "Custom password" and set a secure password.
  - **Password Reset**: Optionally, require the user to reset the password upon first login.



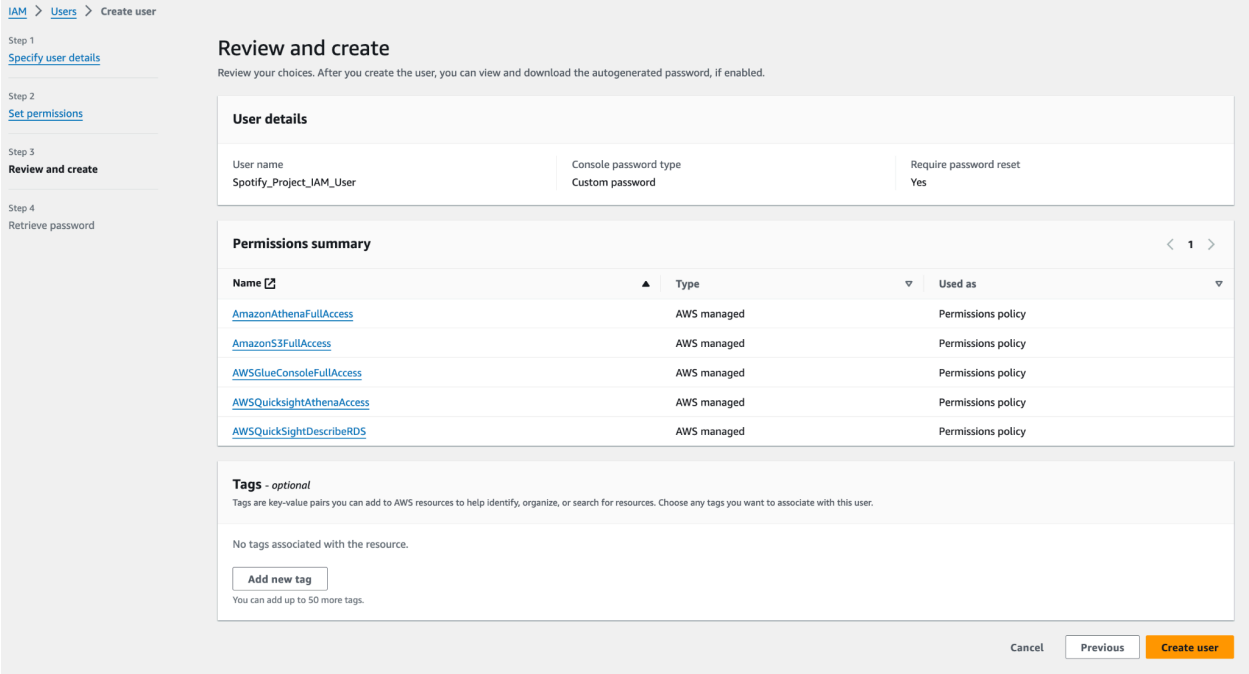
3. **Assign Permissions:**

- On the "Set permissions" page, select "Attach policies directly."



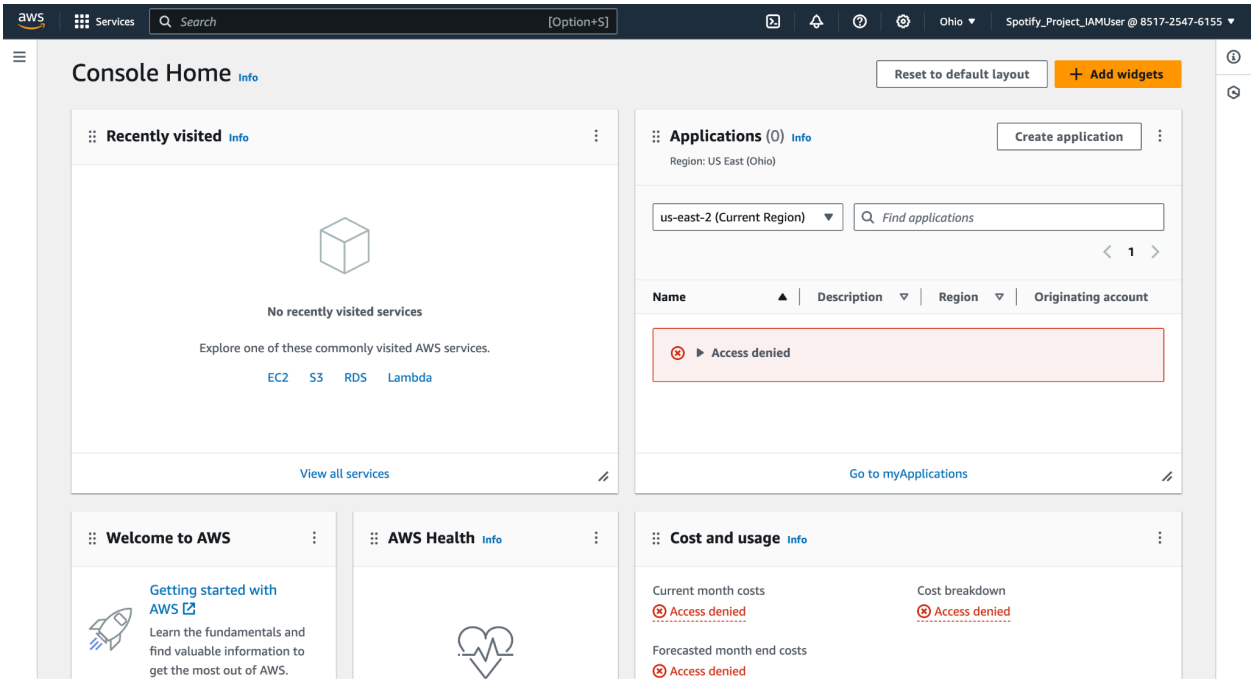
- Search for and select the following policies:
  - AmazonS3FullAccess
  - AWSGlueConsoleFullAccess
  - AmazonAthenaFullAccess
  - AmazonQuickSightFullAccess
  - AWSQuickSightDescribeRDS
  - IAMFullAccess(not included in the screenshot)

- Click "Next" and then "Create user."



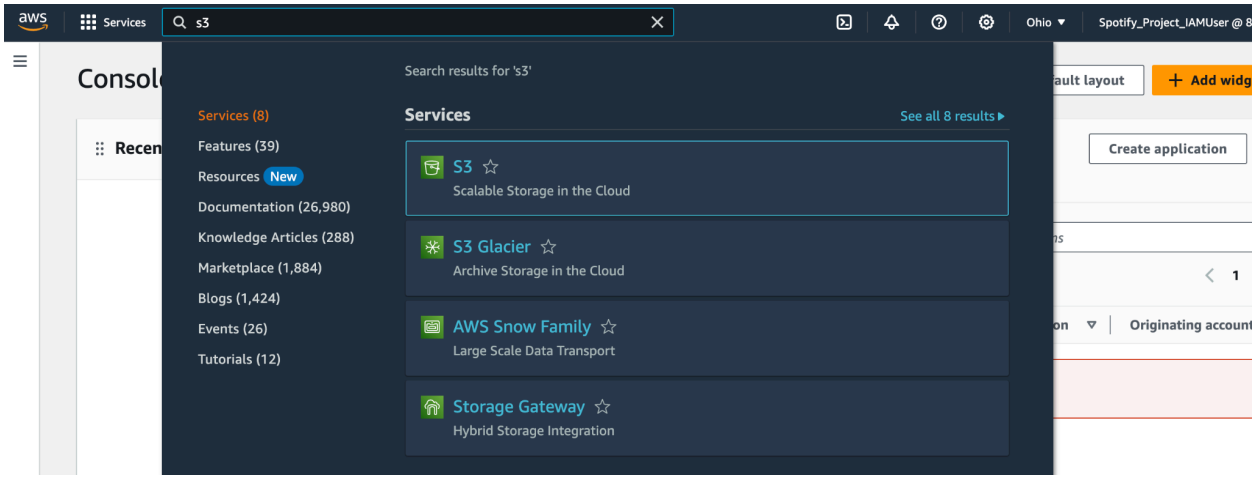
4. **Log in as IAM User:**

- Log out of your root account.
- Log in to the AWS Management Console using the newly created IAM user credentials.

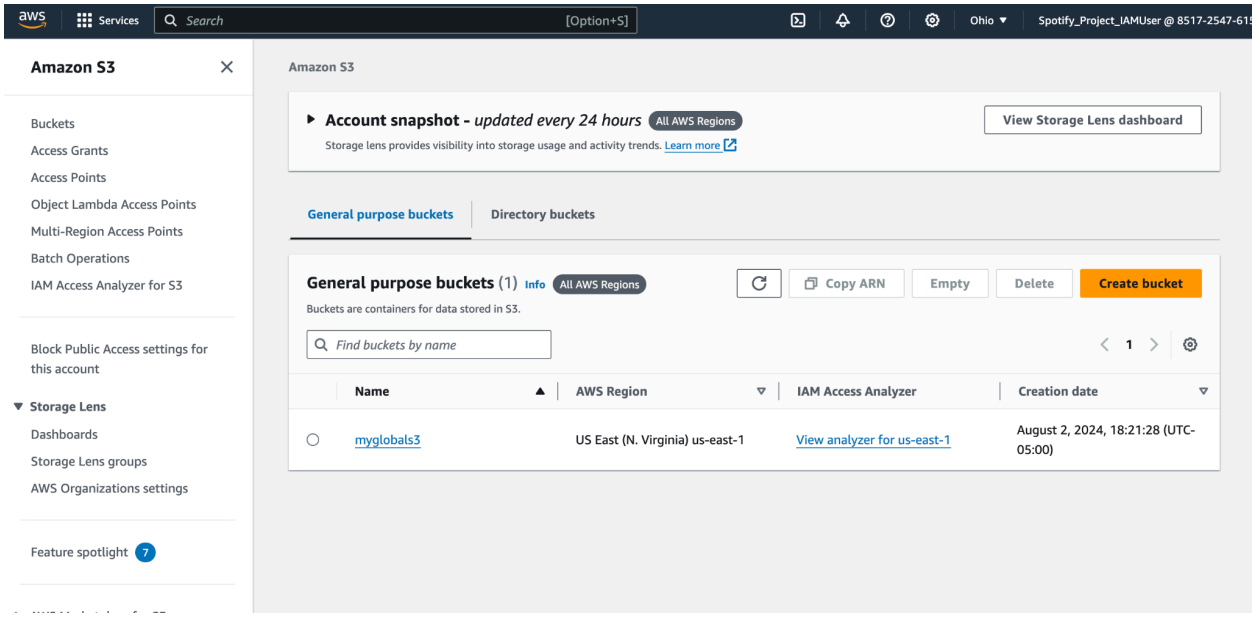


Step 2: Creating S3 Buckets

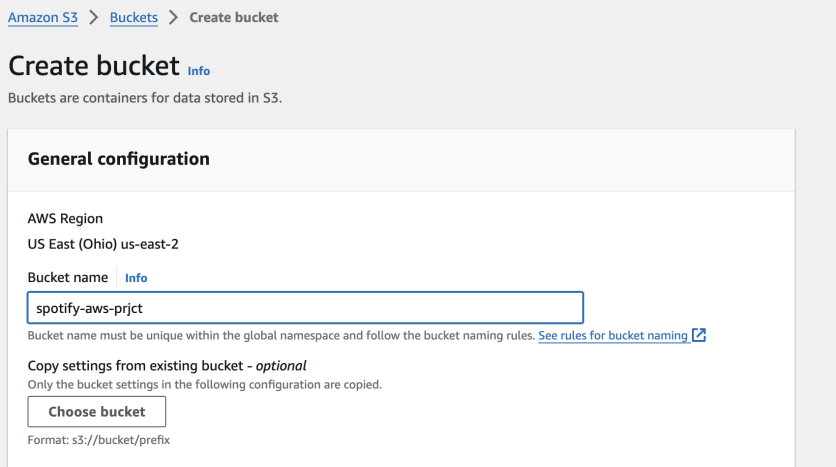
1. **Navigate to S3 Service:**
   - In the AWS Management Console, search for “S3” in the search bar and select the S3 service.



2. **Create a New S3 Bucket:**
   - Click on "Create bucket."



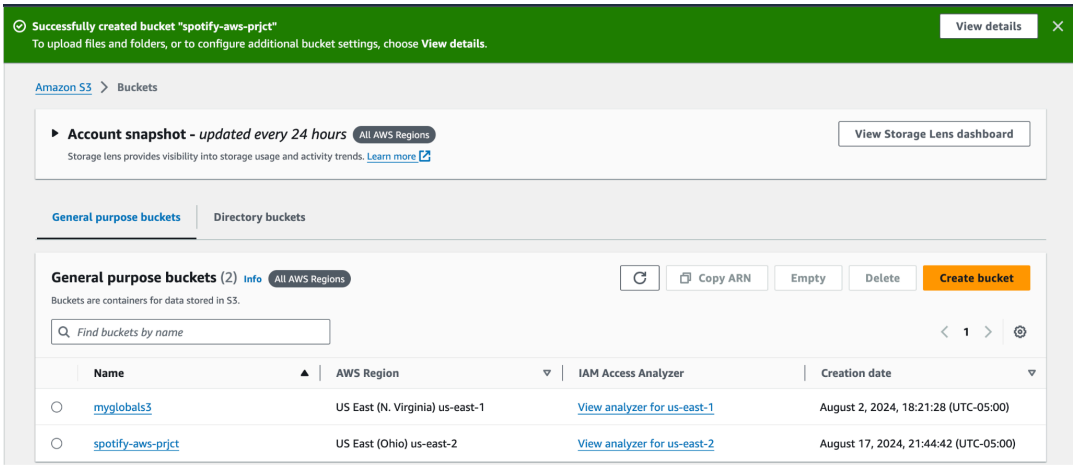
   - **Bucket Name**: Enter {GlobalUnique|NoCaps|NoUnderscore}.
   - **Region**: Select the region closest to you.



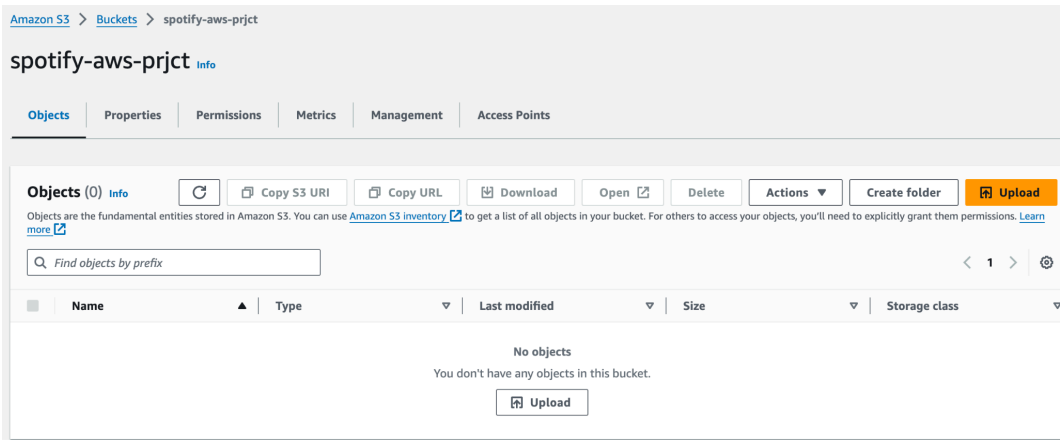
- Leave all other settings as default and scroll down to click "Create bucket."

3. **Create Folders in the S3 Bucket:**

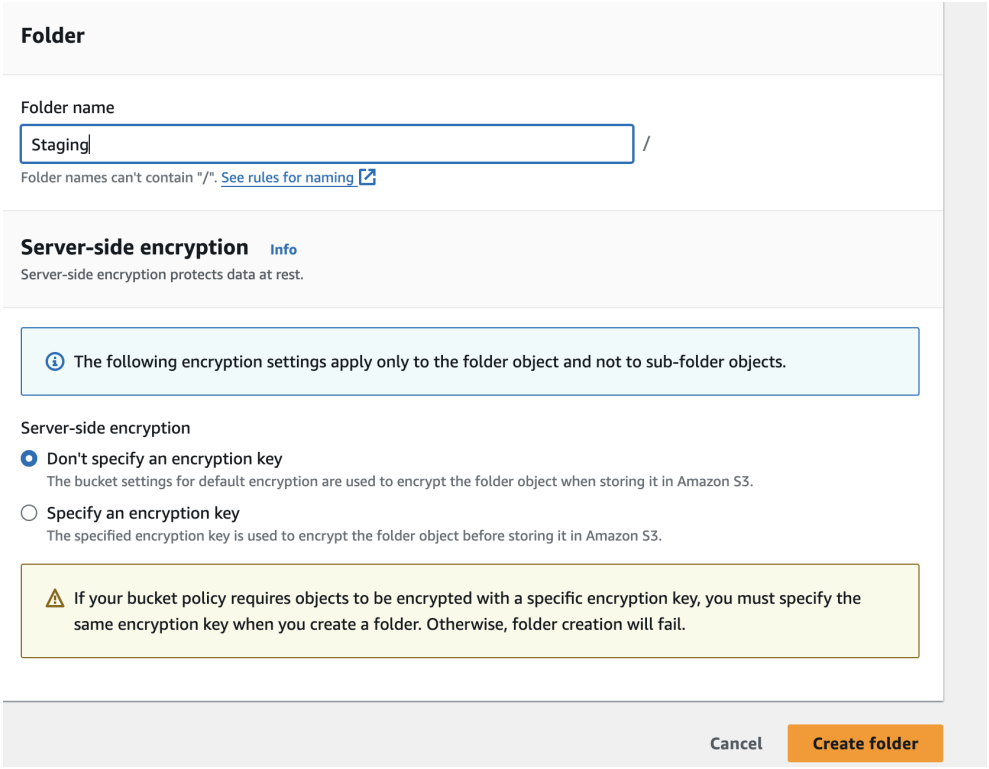
- Click on the bucket project-data you just created.



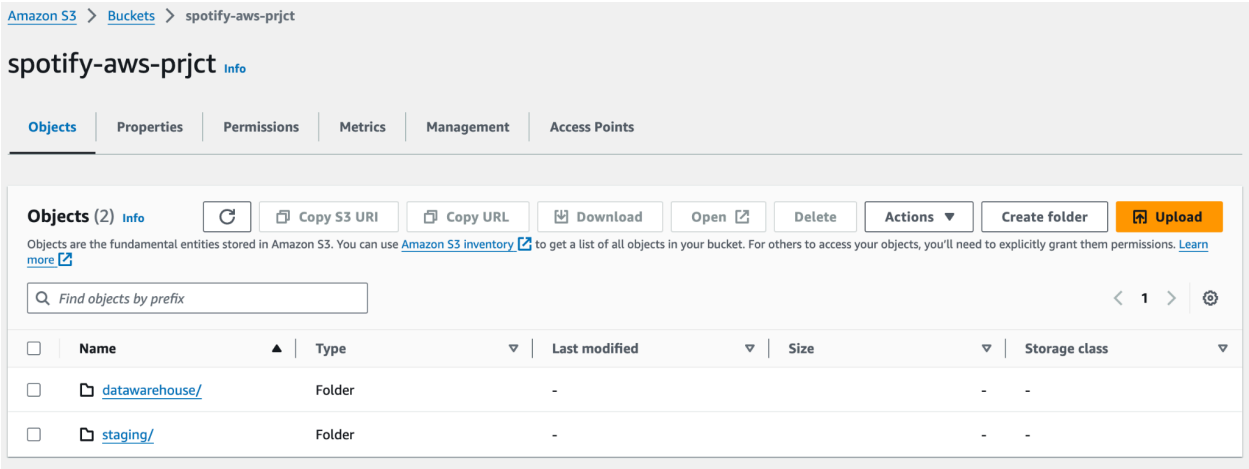
- Click on "Create folder."



- **Folder Name**: Enter staging and click "Create folder."



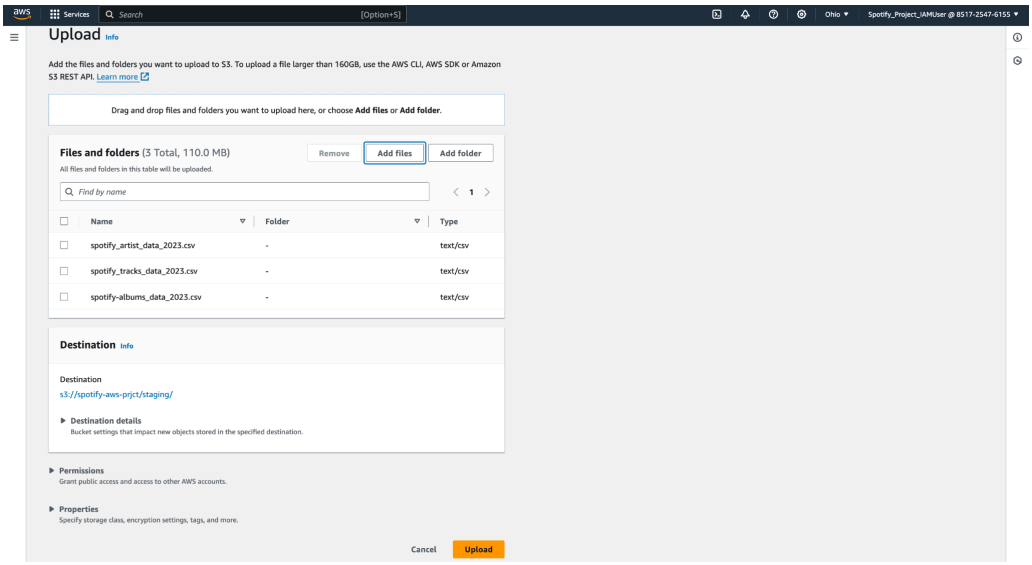
- Repeat the above step to create another folder named data-warehouse.



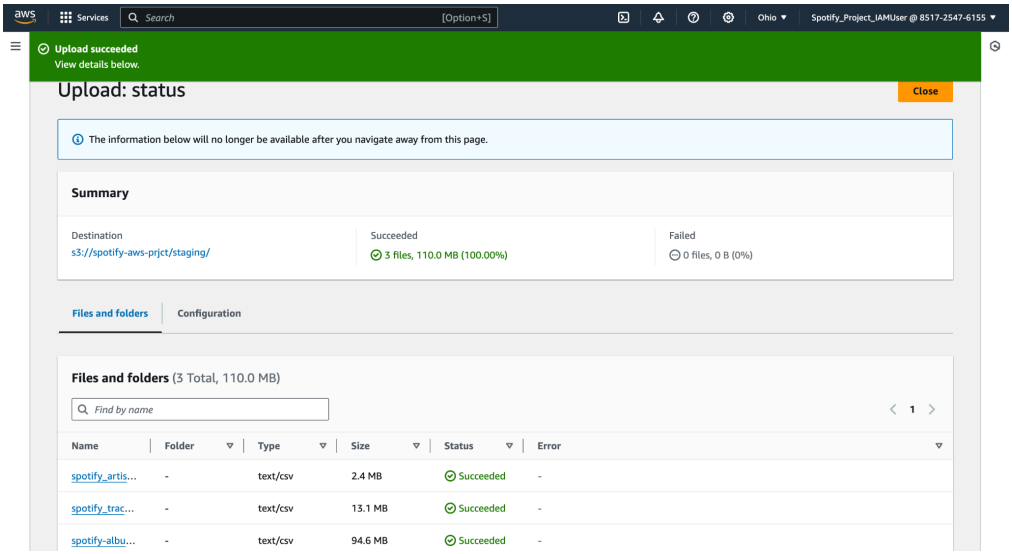
4. **Upload Pre-Processed CSV Files:**

In real-time data in a staging layer will be coming from through Dynamo DB or our database instance, but for this project, we are not making use of Dynamo DB or database hence we are adding our data manually

- Click on the staging folder.
- Click "Upload" and then "Add files."
- Select the pre-processed CSV files (albums.csv, artists.csv, tracks.csv) from your local machine.



- Click "Upload" to upload the files to the staging folder.

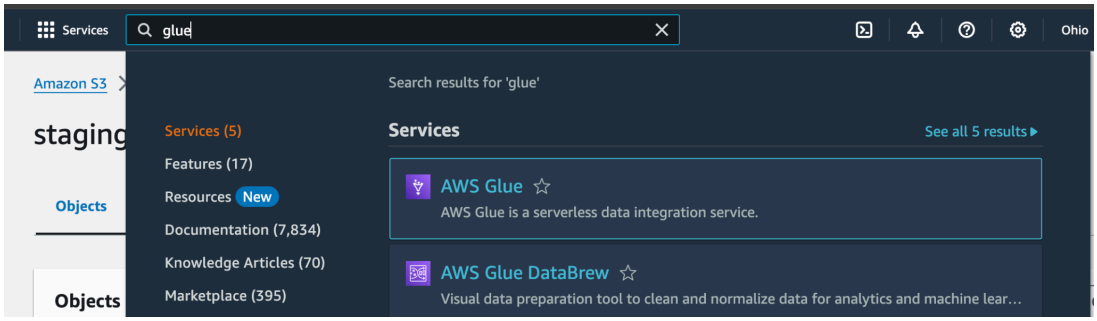


Step 3: Setting Up AWS Glue for ETL

In this step, we'll try to create a data pipeline that will transfer our data from the staging layer to the data warehouse. we'll be making use of AWS glue to create a data pipeline. AWS glue is a managed service provided by AWS to create a data pipeline.

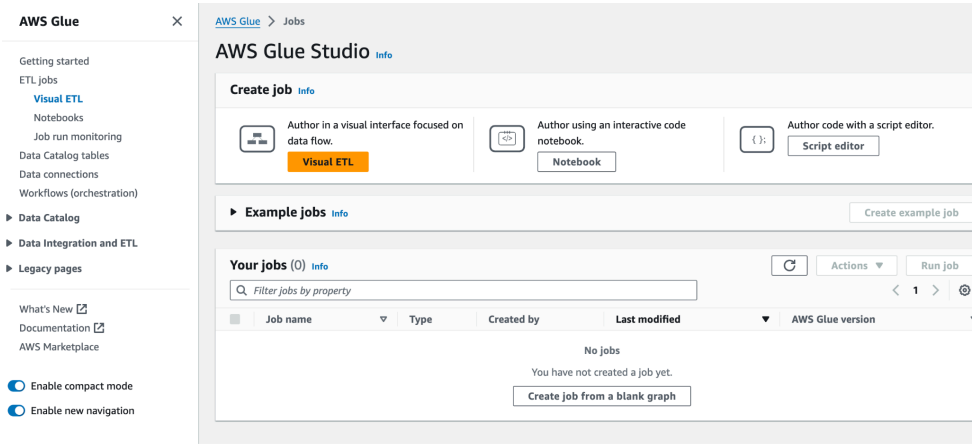
1. **Navigate to AWS Glue Service:**

- In the AWS Management Console, search for “Glue” in the search bar and select the AWS Glue service.



2. **Create a New Glue Job:**

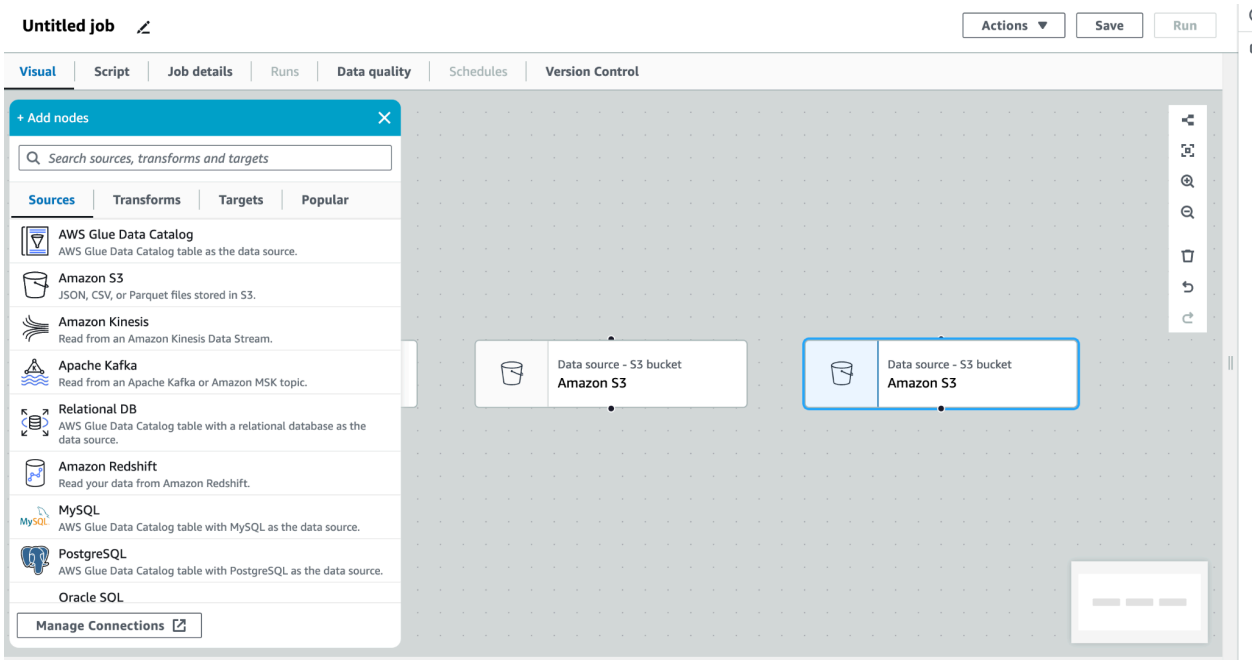
- In the AWS Glue dashboard, click on "Visual ETL" under the "ETL Jobs" section.



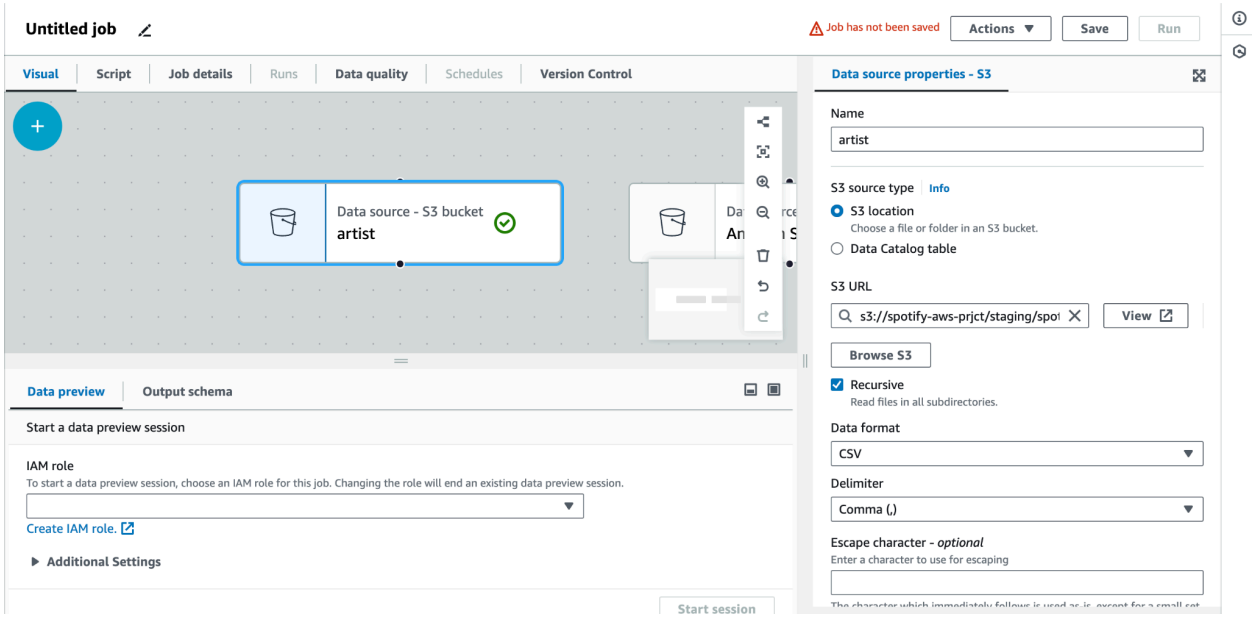
- Click "Visual ETL." under Create job.

3. **Set Up Data Sources:**

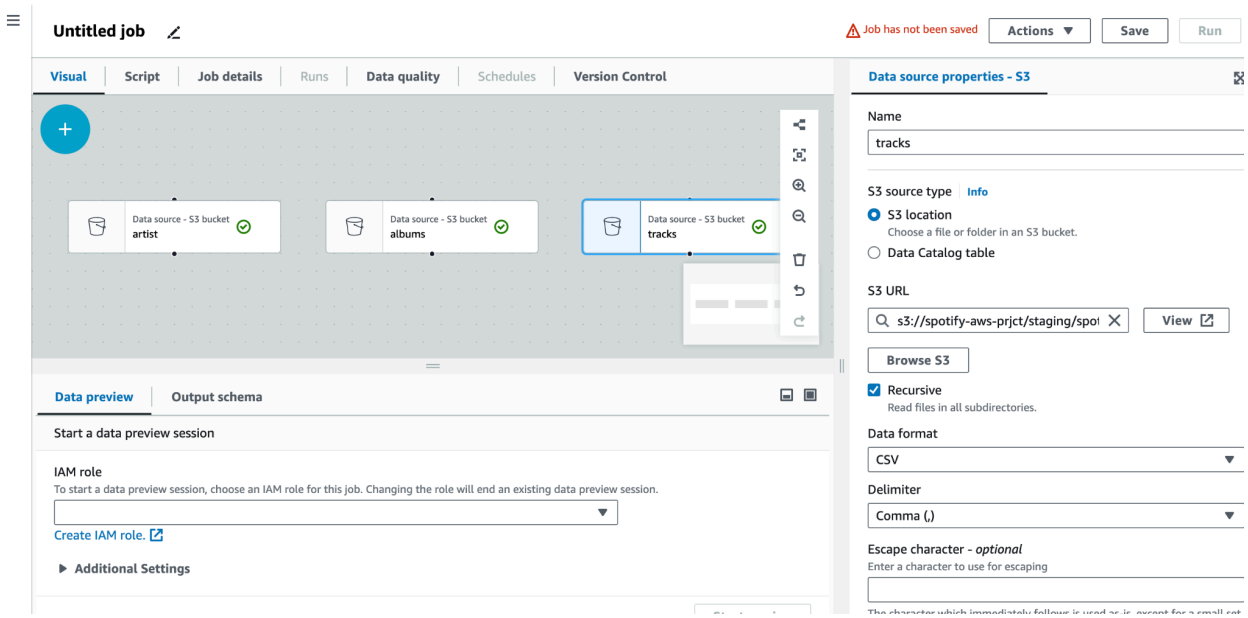
- Drag and drop source and destination S3 buckets, as we have three CSV files (albums.csv, artists.csv, tracks.csv)



- Select the First Amazon S3 bucket and rename it with “artist” and click on browse and select the file from the **s3://spotify-aws-prjct/staging/spotify_artist_data_2023.csv.** Select the Data format as CSV.

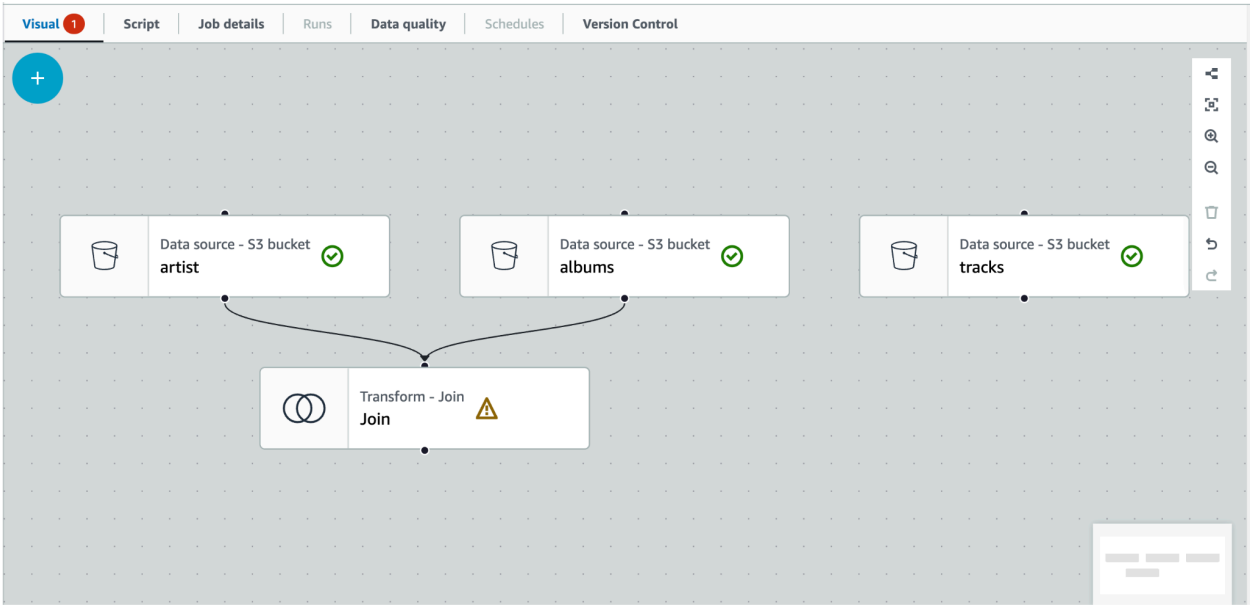


- Repeat the same steps for the other 2 S3 Buckets.

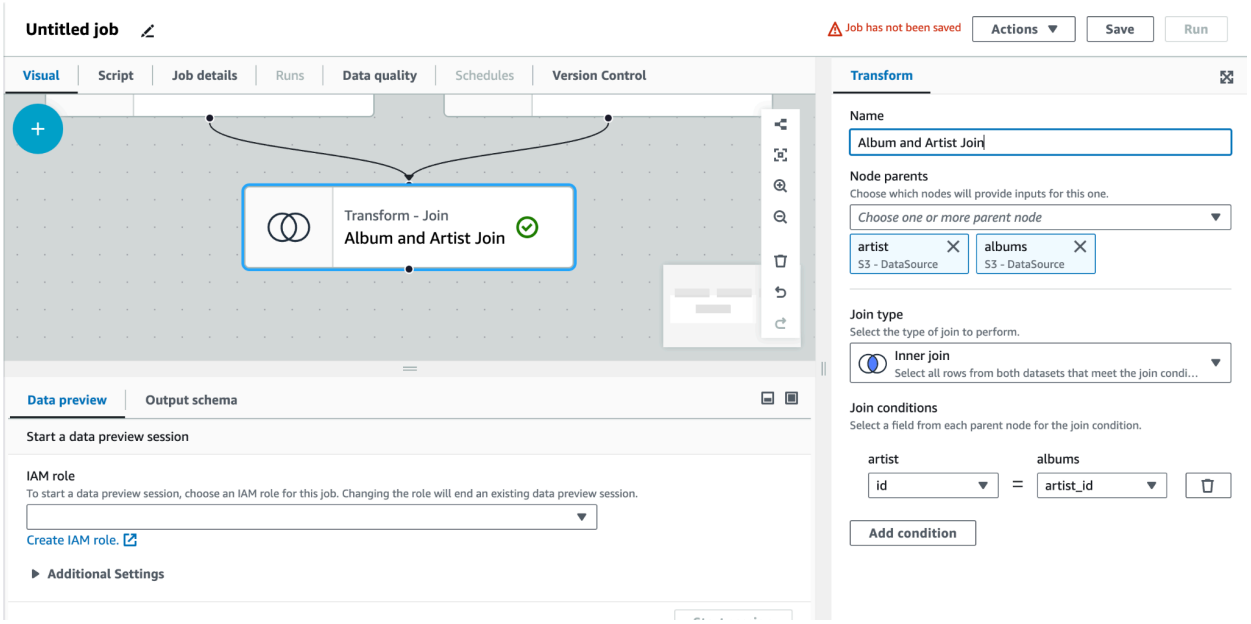


4. **Configure Data Transformations:**

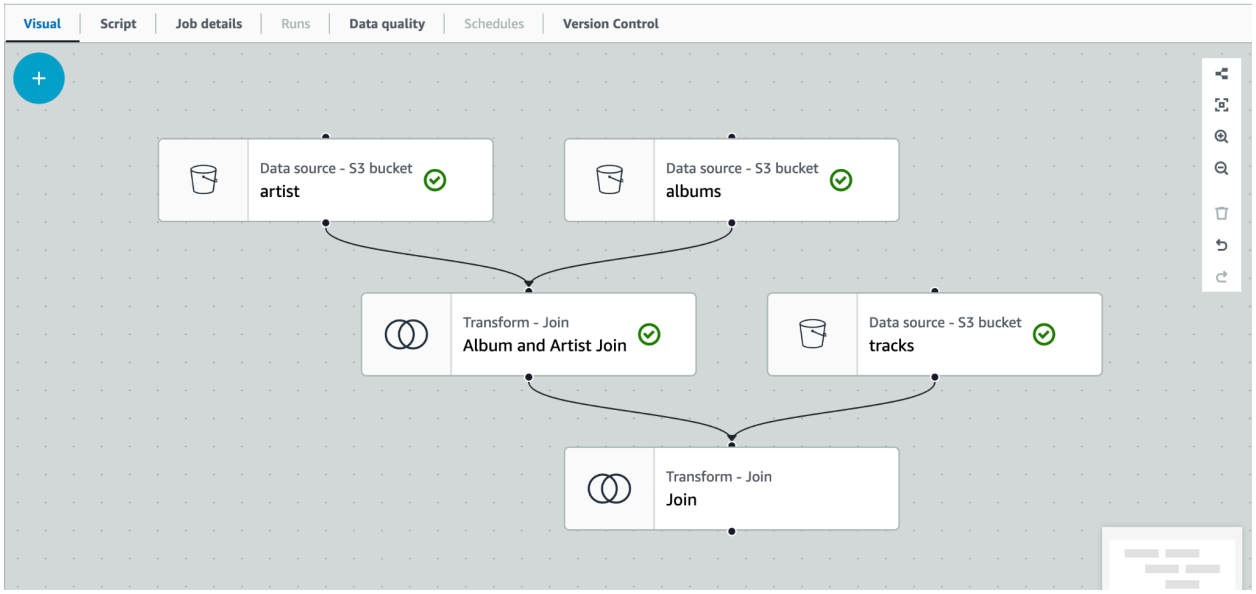
- Now to join album and artist, click on the ADD symbol, select join from **Transforms** and connect nodes as per the image below.



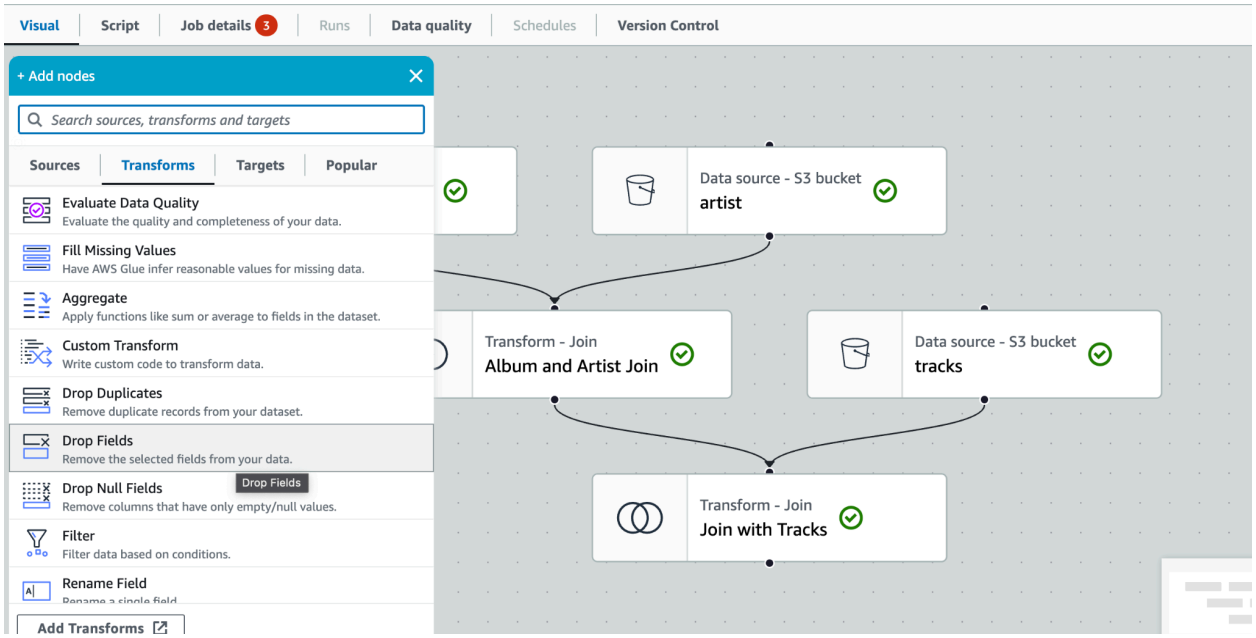
- After Joining the nodes of both the buckets to the Join Transform. Add a Condition where *artist 'id' = albums 'artist_id'* and rename the join **"Album and Artist Join".**



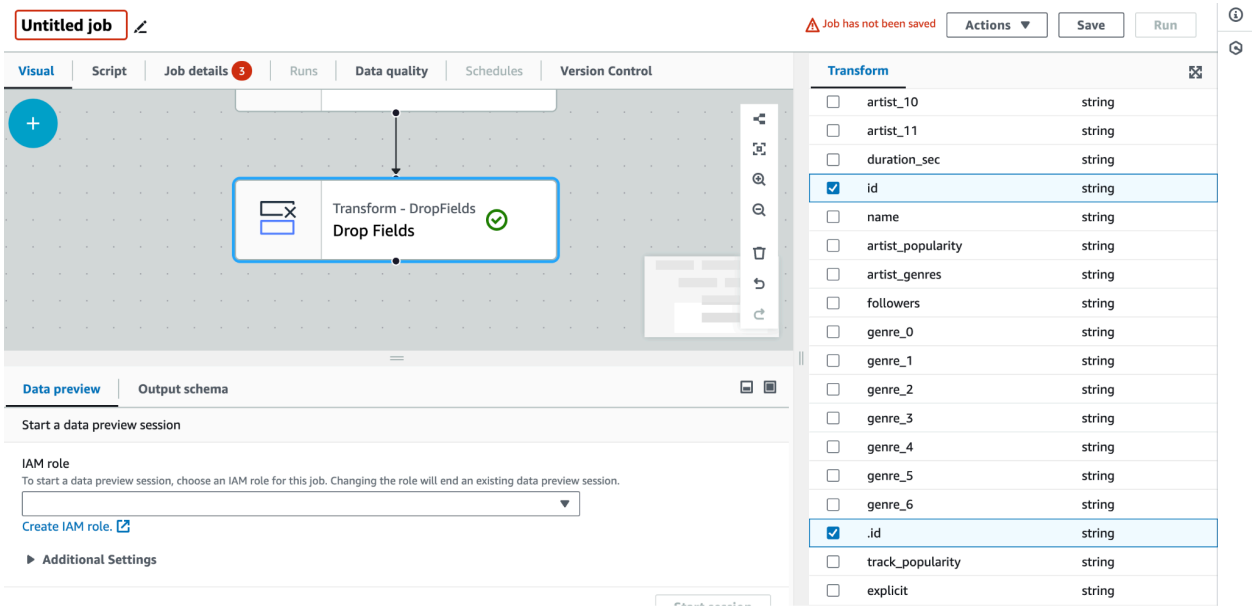
- Now add another Join Transfrom to join '**track'** s3 bucket and '**Album and Artist'** Join



- Now select the Join and add the condition *Album and Artist Join 'track_id' = tracks 'id'*
- and rename the join as 'Join with Tracks'.
- To drop unnecessary columns, select **Drop Fields** from Transforms node.

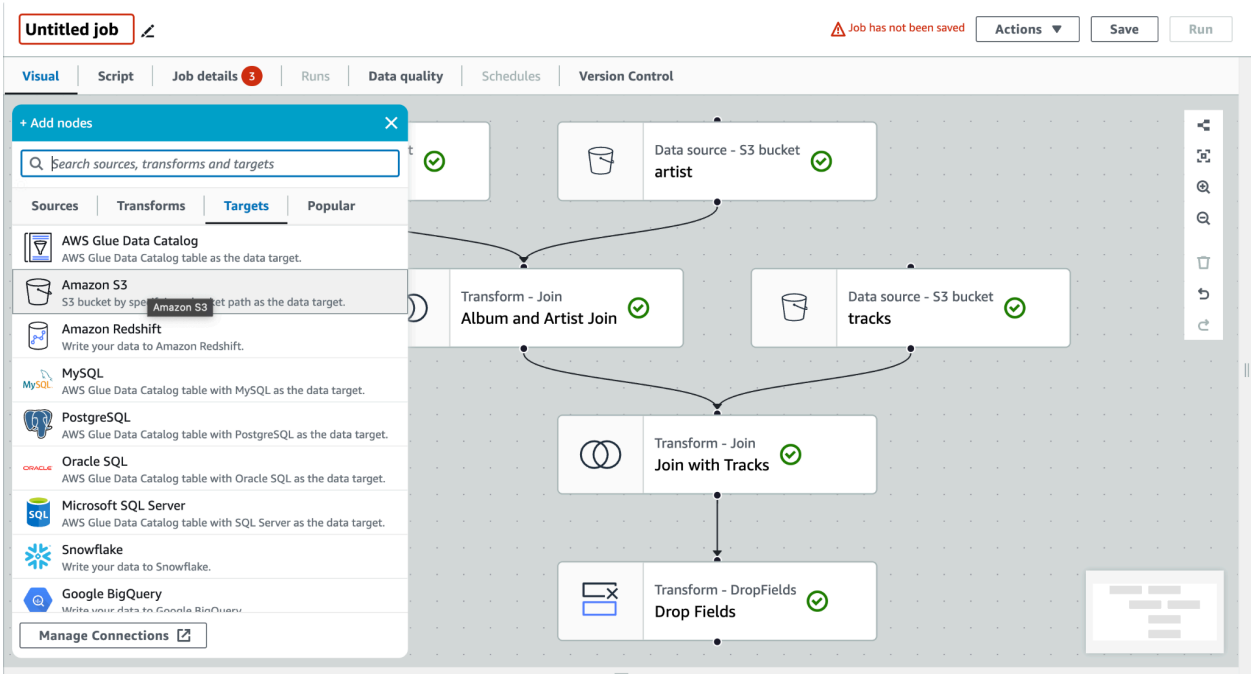


- We remove duplicate columns and Identical columns we dont need. Here id is inner join to artist id so select id.

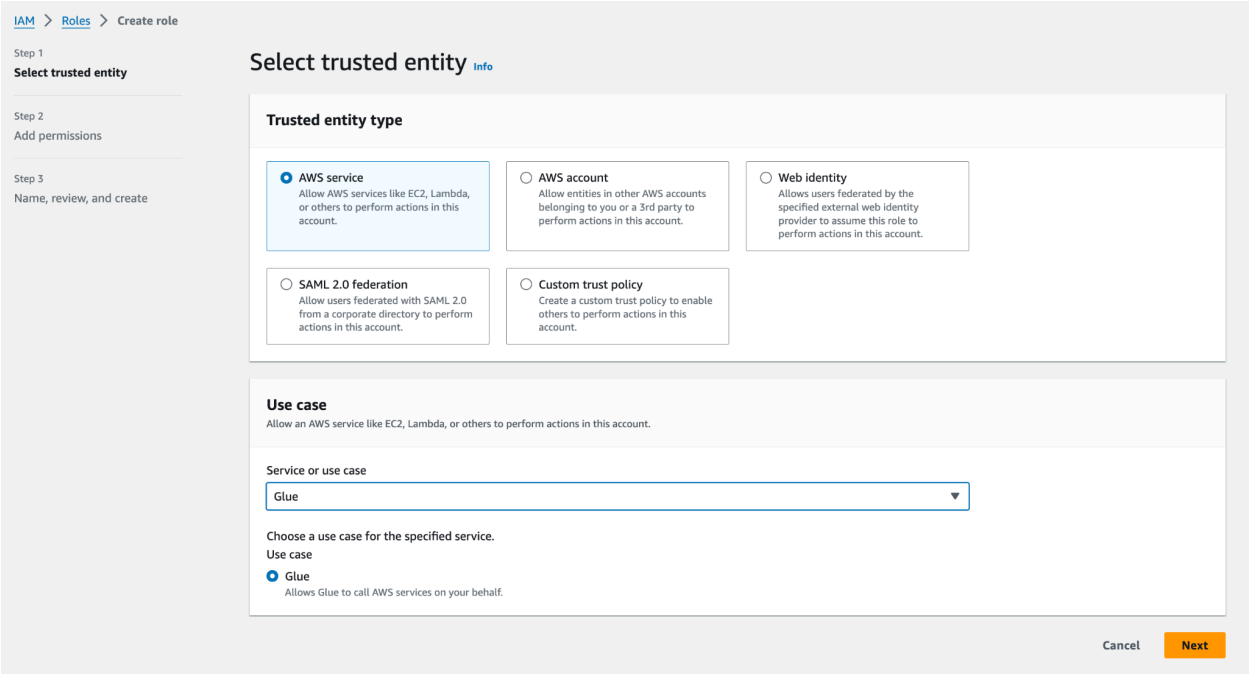


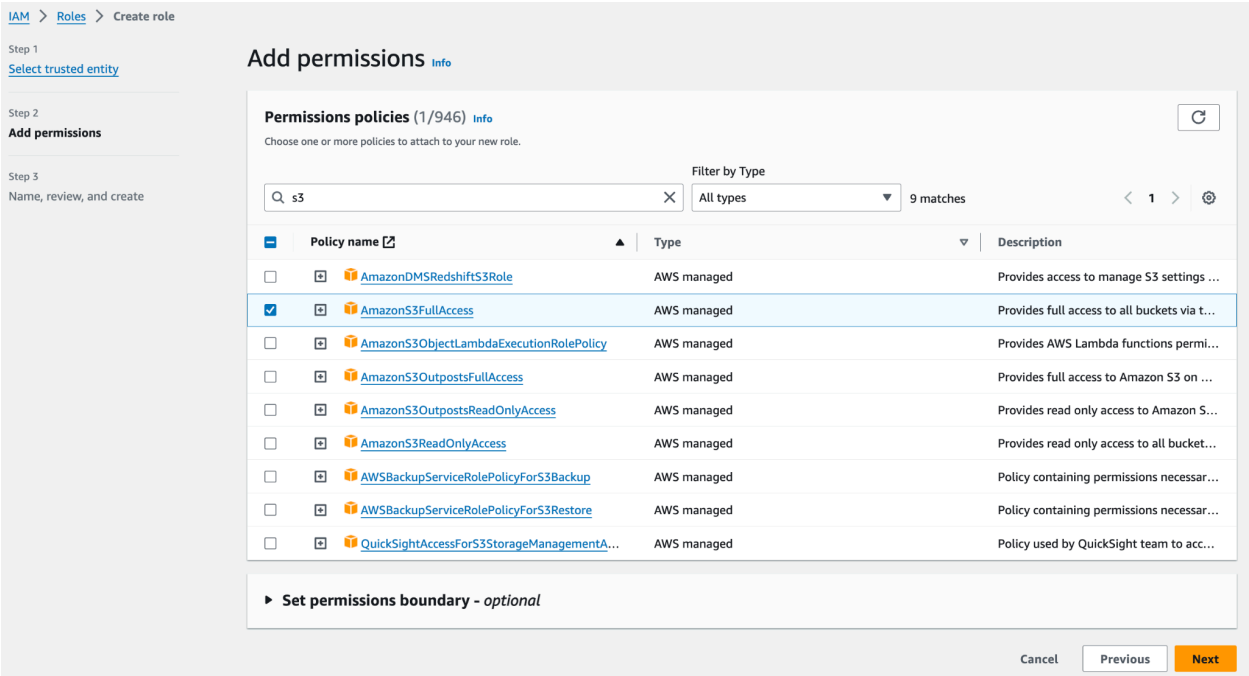
5. **Set Up Data Target:**

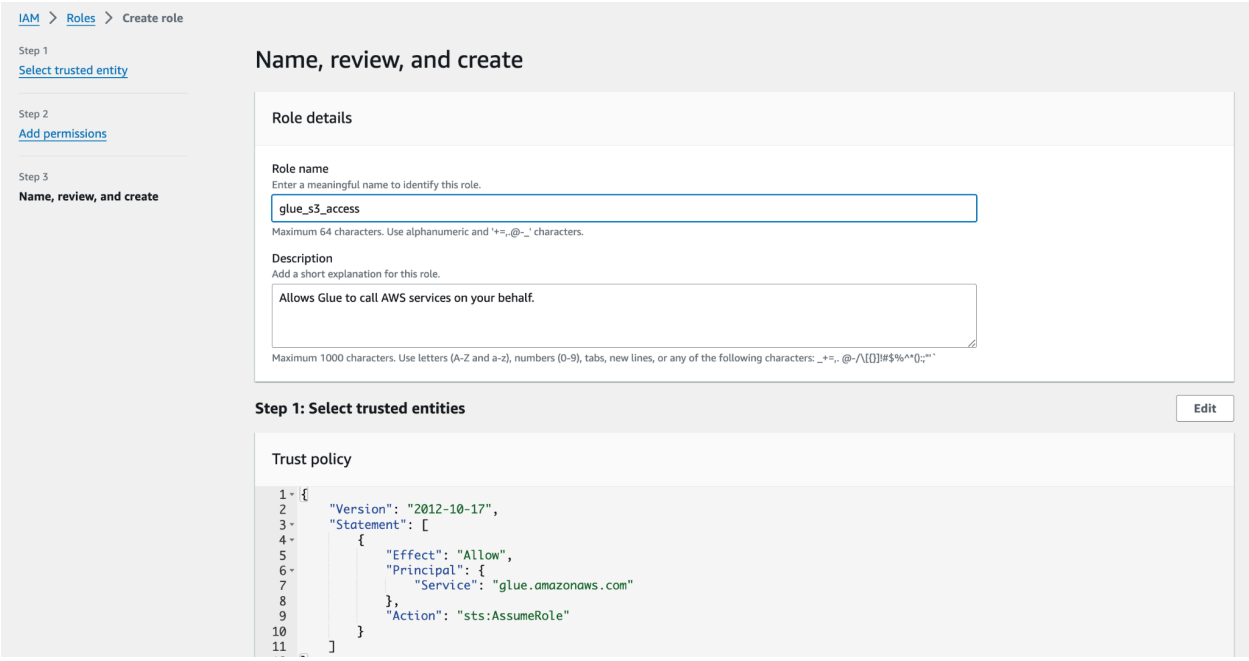
- Next step is to add the destination. Select Amazon S3 bucket in the Targets Section and add



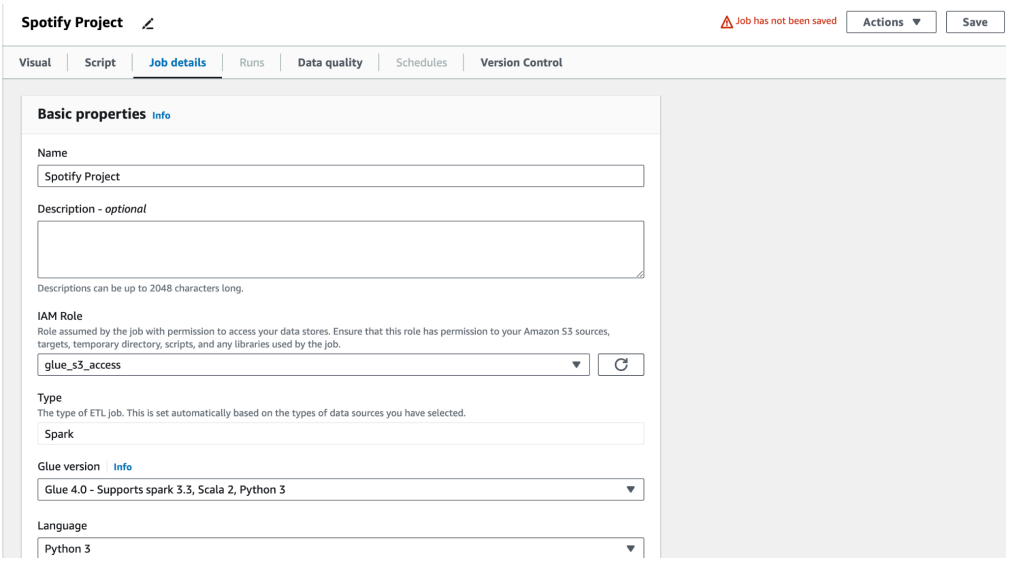
- Rename the Destination node and add the Target location "s3://spotify-aws-prjct/datawarehouse/" and make sure the compression type is Snappy.
- Add the **Job Name**: Enter a name like Spotify Project.
- As there is No IAM role. Login to root user and create an IAM role using below steps in the screenshots.





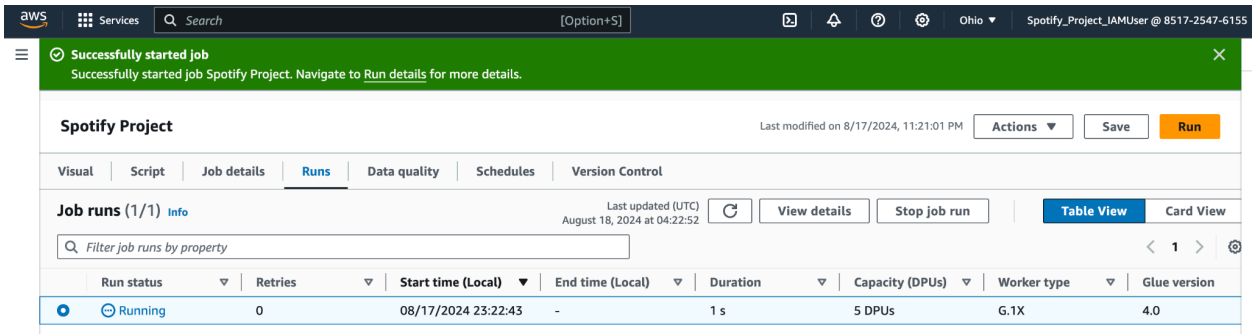


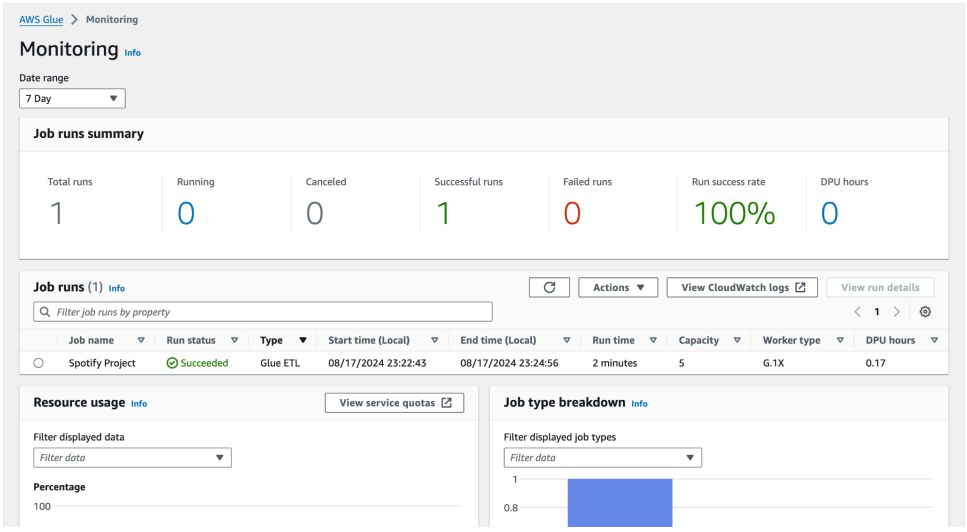
- **IAM Role**: Select the IAM role that has the required permissions.
- **Type**: Choose "Spark."
- **Glue Version**: Choose the latest Glue version available.
- **Python Version**: Select Python 3.
- **Script Path**: Leave the default script path or specify a path in your S3 bucket.
- Click **Save**



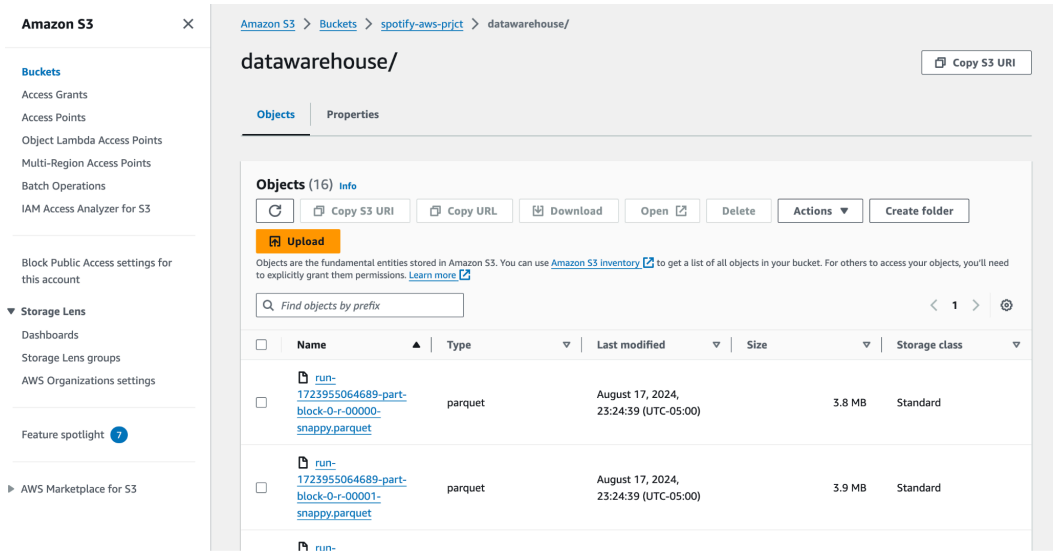
6. Run the Glue Job:

- Review your job settings.
- Click "Run job" to start the ETL process, transforming and moving data from the staging layer to the data warehouse.





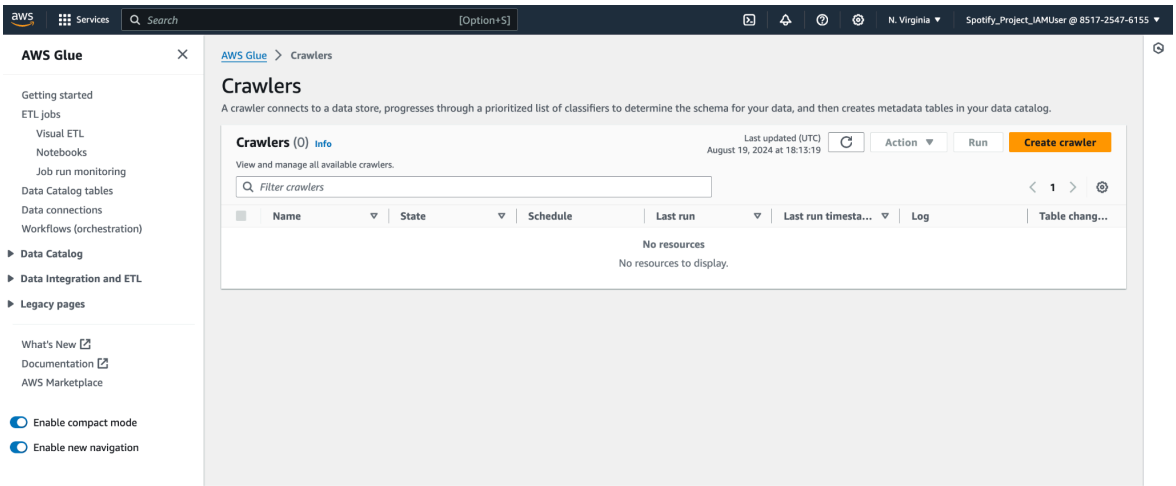
- Check S3 bucket if the files are Parsed



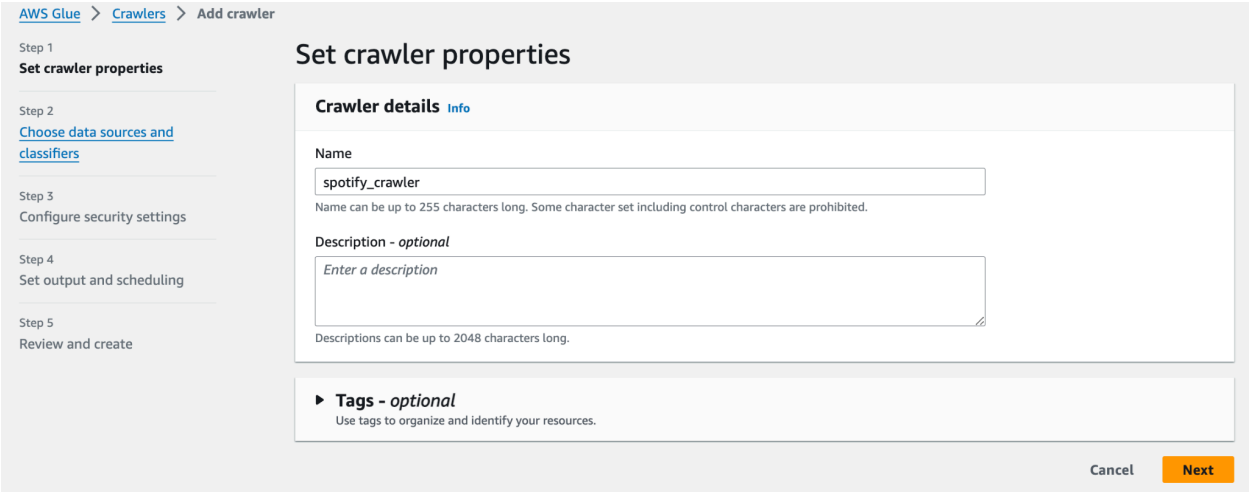
Step 4: Creating a Data Catalog with AWS Glue Crawler

1. **Create a New Crawler:**

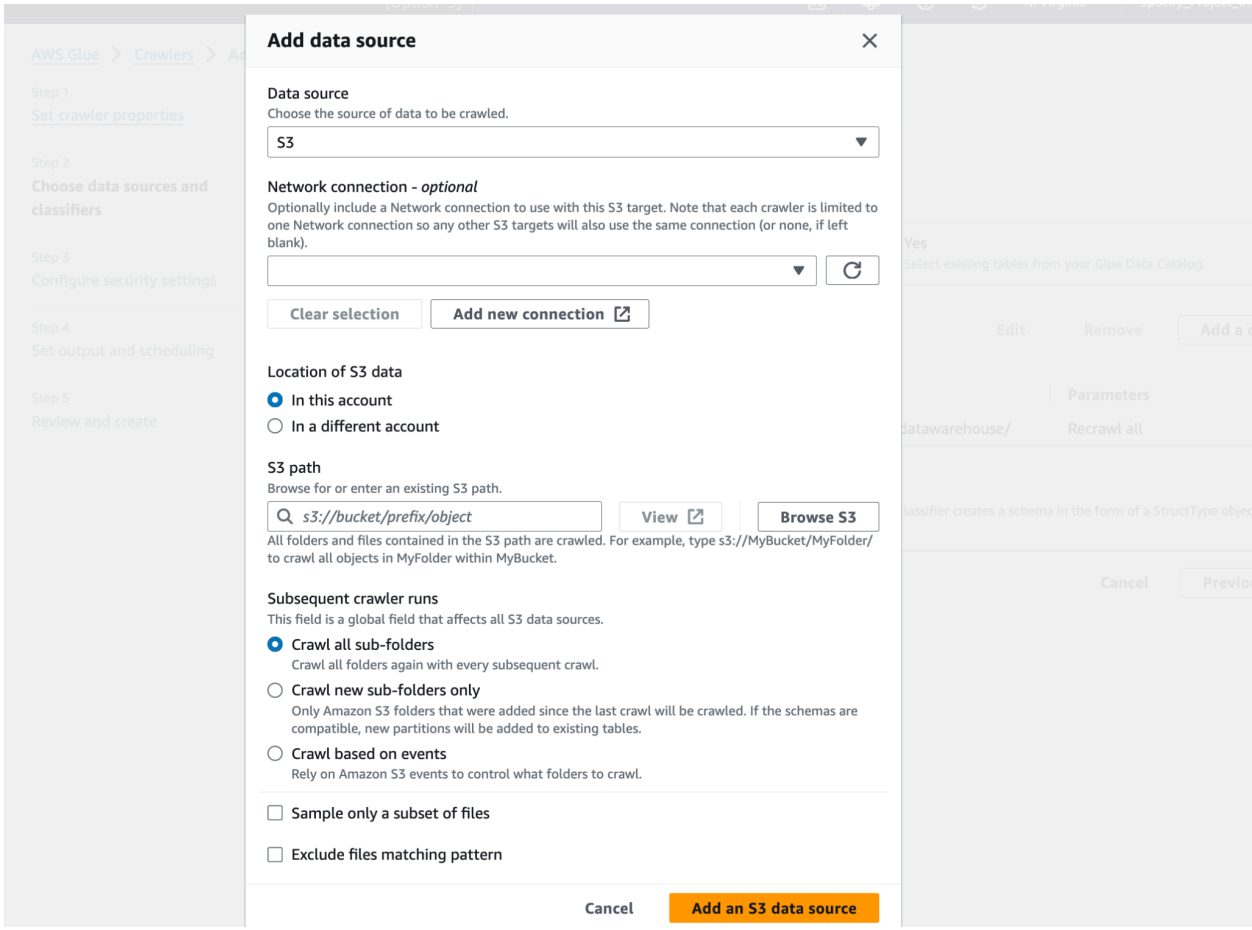
- In the AWS Glue dashboard, click on "Crawlers" under the "Data catalog" section.
- Click "Create crawler."

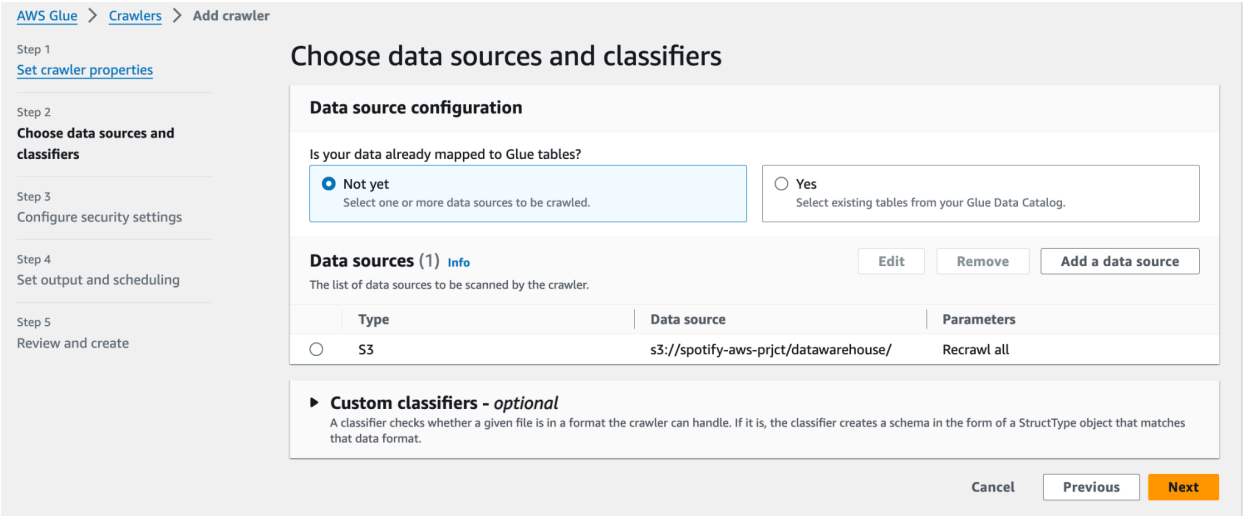


- **Crawler Name**: Enter spotify_crawler.

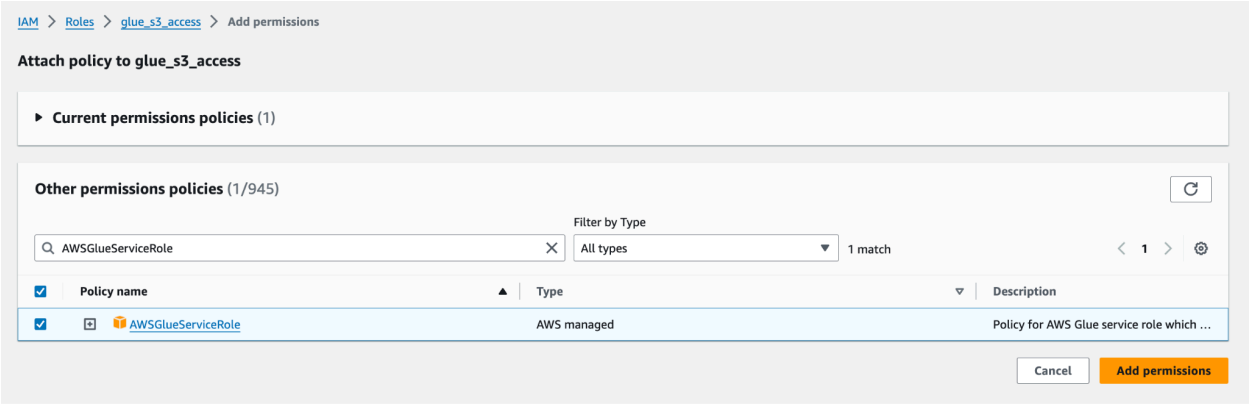


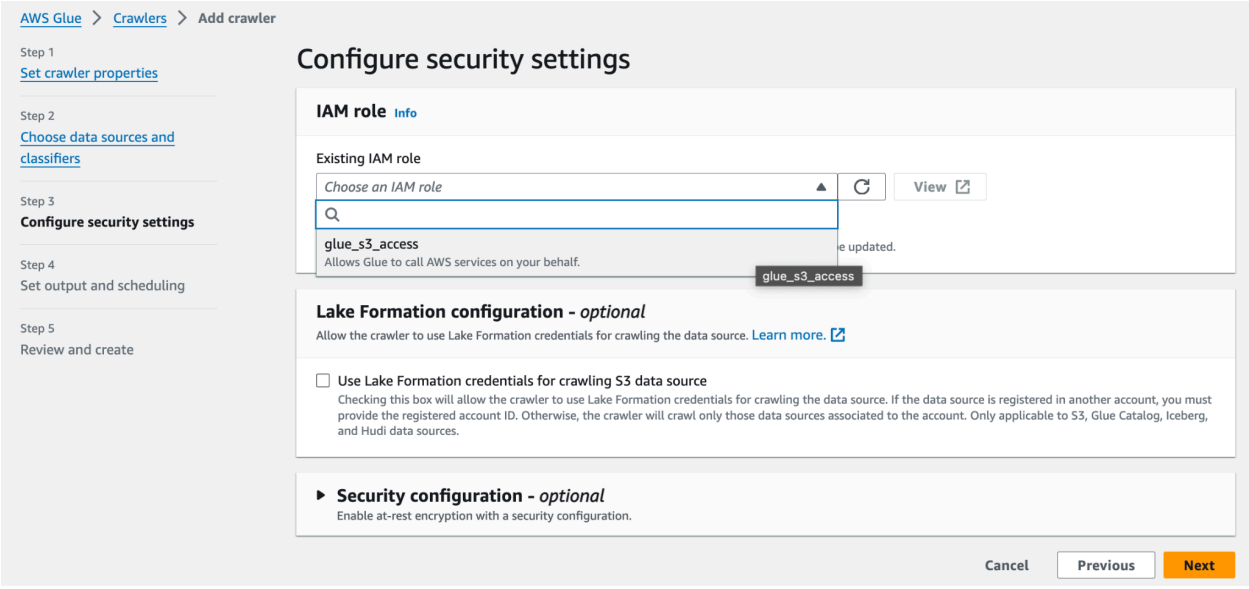
- **Data Store**: Select S3 and provide the path to the data-warehouse folder.





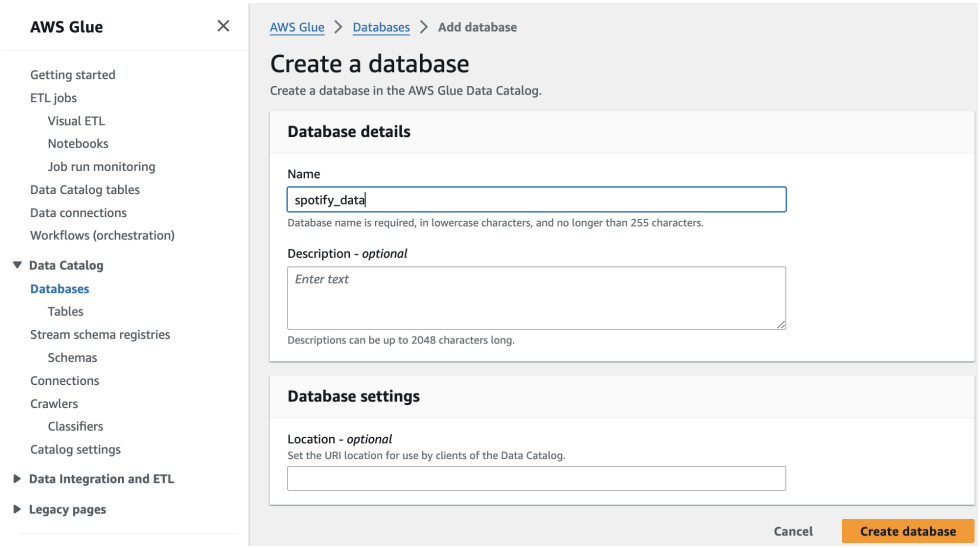
- **IAM Role:** Select the IAM role we created earlier and before this please do add another policy "AWSGlueServiceRole" to this role.



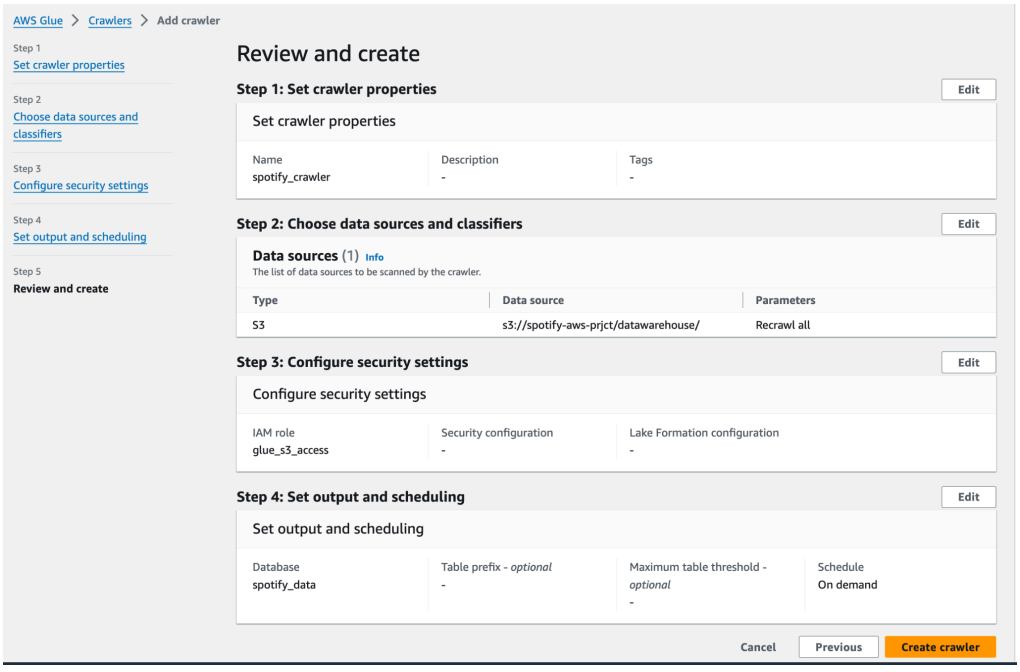


2. **Create a New Database:**

- Open the duplicate tab. Go to the AWS Glue > Data catalog >> Databases, and create a new database.

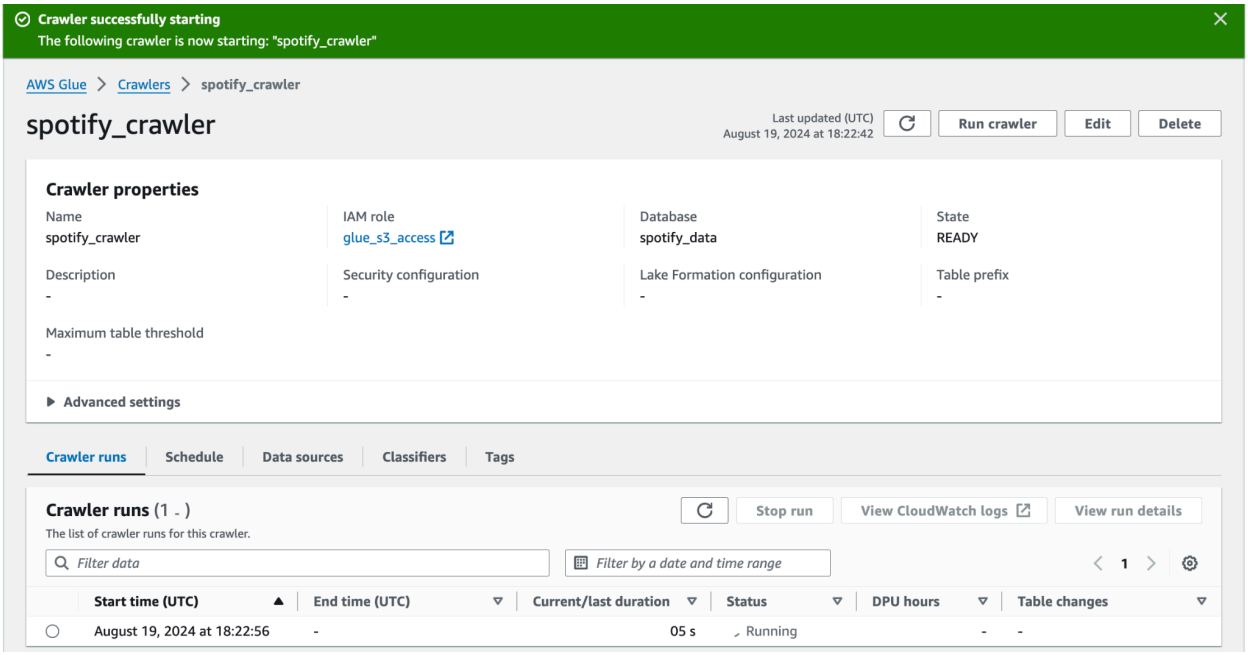


- **Database Name**: Enter spotify_data.
- Back to the Crawlers, select the created database
- Click Next and Create the Crawler.

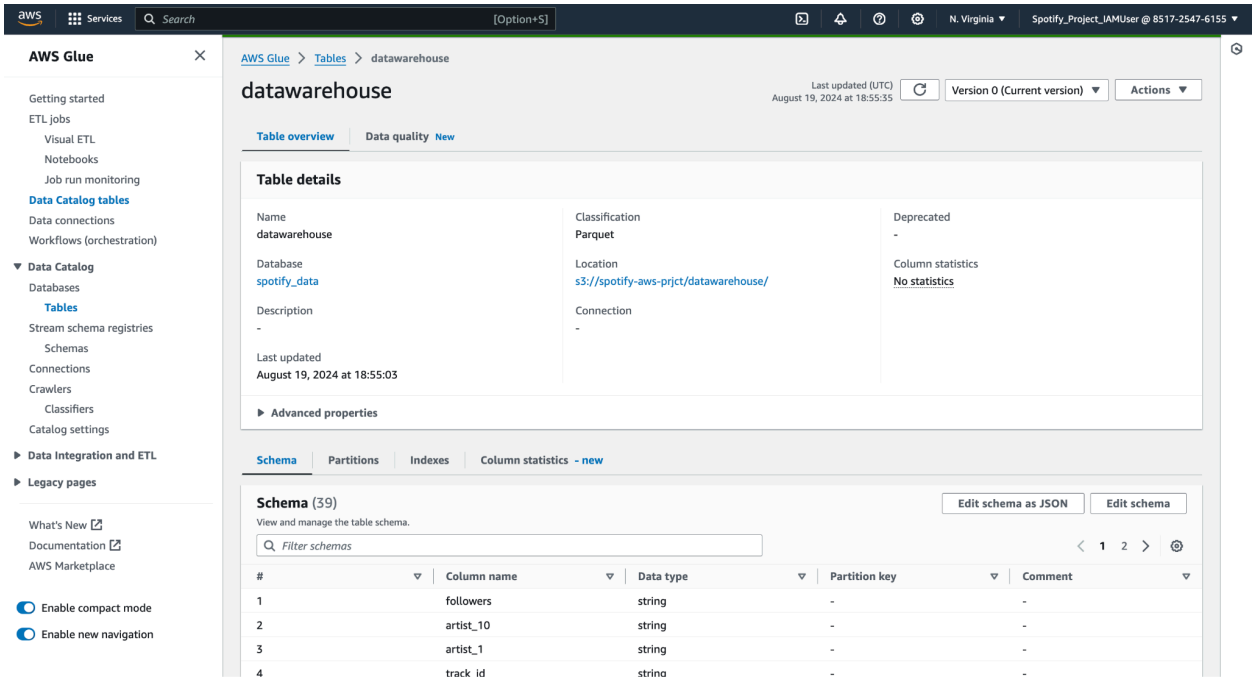


3. **Run the Crawler:**

- After setting up the crawler, click "Run crawler."



- The crawler will scan the data in the data-warehouse folder and create corresponding tables in the spotify_data database.

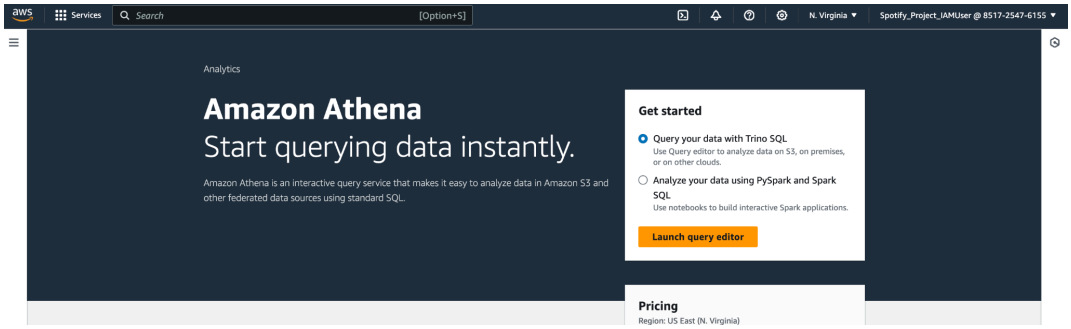


Step 5: Querying Data with AWS Athena

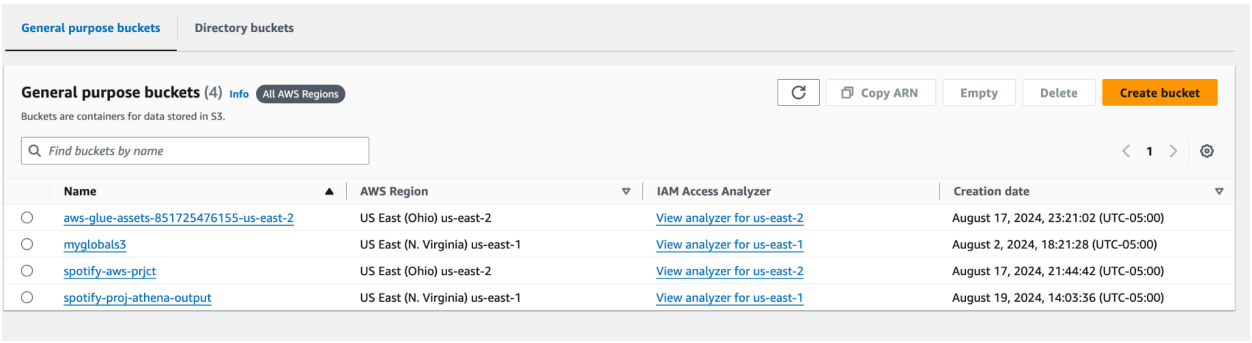
1. **Set Up Query Result Storage:**

- Navigate to AWS Athena from the AWS Management Console.

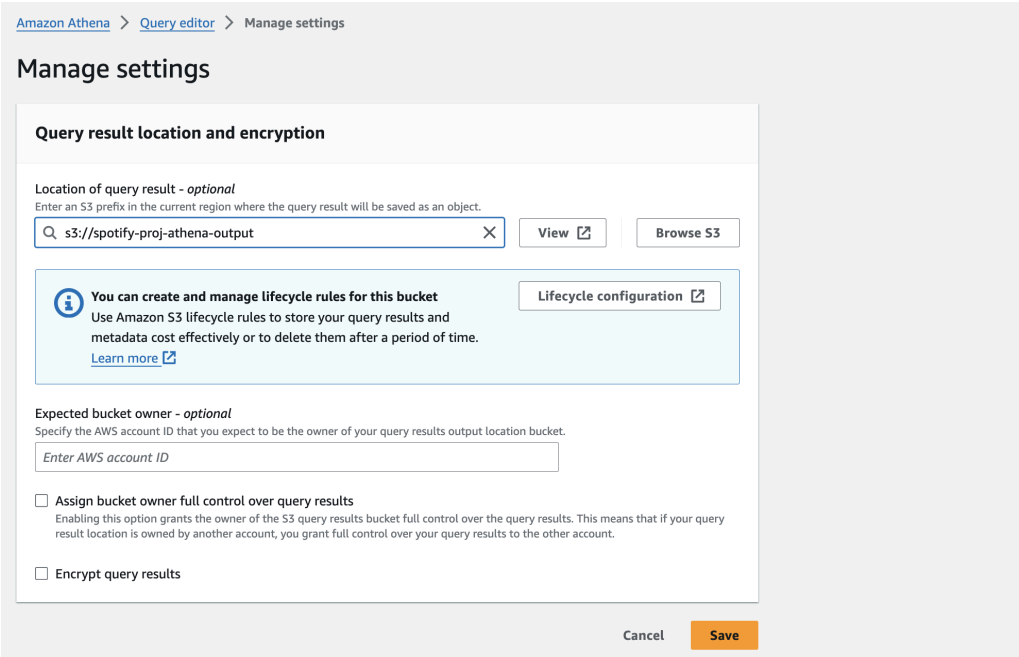
- Click on ‘Launch query editor'



- Before running any queries, you must specify a location for query results.
- Create a new S3 bucket named spotify-proj-athena-output.



- In the Athena settings, set the query result location to this new bucket.



2. **Write SQL Queries:**

- In the Athena query editor, write SQL queries to analyze the data.

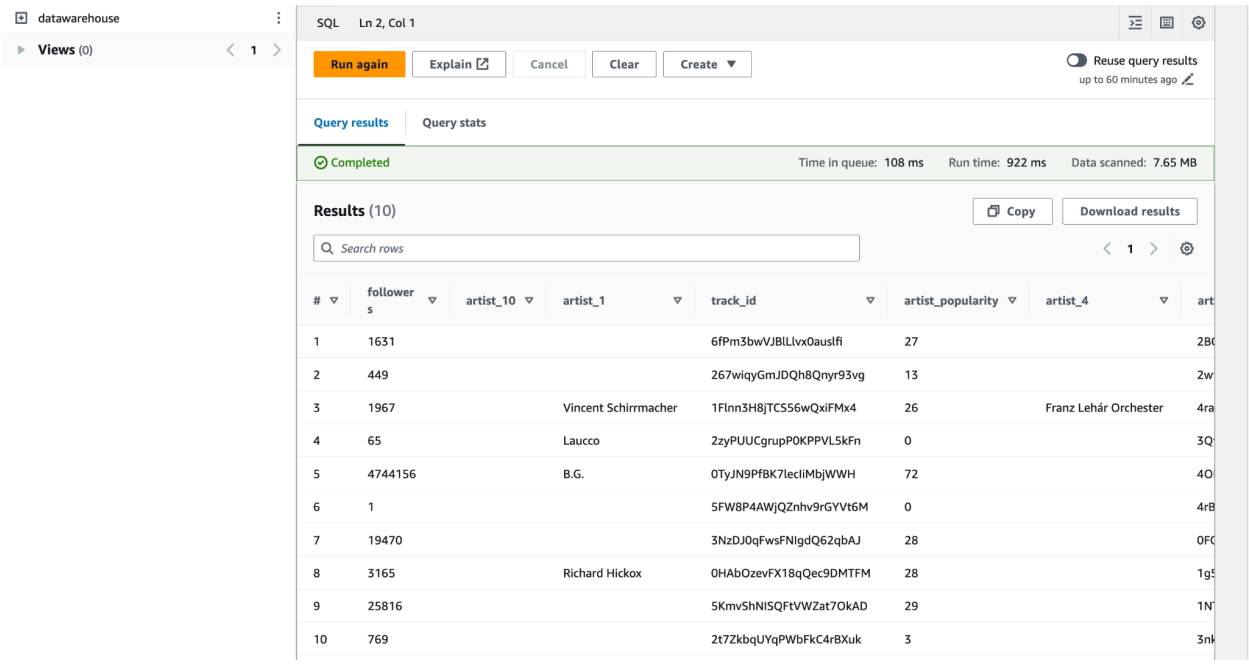
Example Query:

SELECT * FROM datawarehouse LIMIT 10;

- This query will fetch the first 10 records from the data_warehouse table.

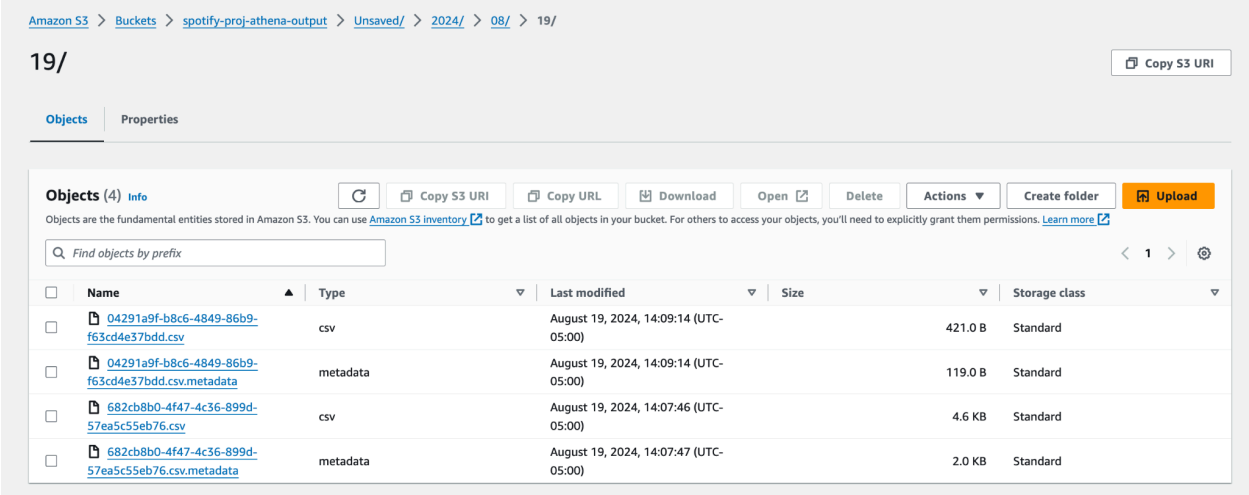
3. **Run the Query:**

- Click "Run query" to execute the SQL statement.



#	followers	artist_10	artist_1	track_id	artist_popularity	artist_4	art
1	1631			6fPm3bwVJBlLlvx0auslfi	27		2B
2	449			267wiqyGmJDQh8Qnyr93vg	13		2w
3	1967		Vincent Schirrmacher	1Flnn3H8jTCS56wQxiFMx4	26	Franz Lehár Orchester	4ra
4	65		Laucco	2zyPUUCgrupP0KPPVL5kFn	0		3Q
5	4744156		B.G.	0TyJN9PfBK7lecliMbjWWH	72		4O
6	1			5FW8P4AWjQZnhv9rGYVt6M	0		4rB
7	19470			3NzDJ0qFwsFNIgdQ62qbAJ	28		0FC
8	3165		Richard Hickox	0HAbOzevFX18qQec9DMTFM	28		1g
9	25816			5KmvShNISQFtVWZat7OkAD	29		1N
10	769			2t7ZkbqUYqPWbFkC4rBXuk	3		3nk

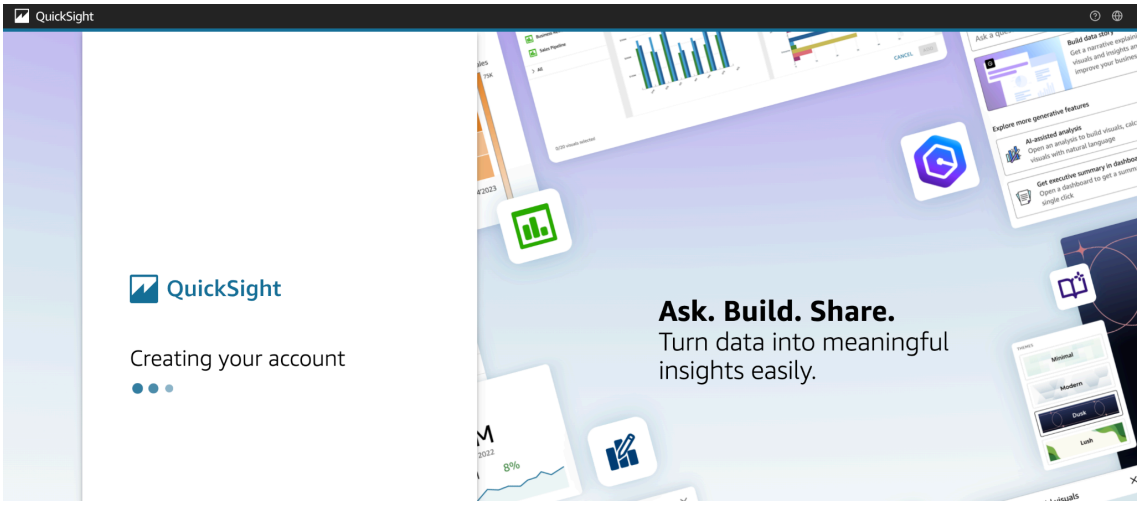
- The results will be displayed in the query editor, and the output will be saved in the specified S3 bucket (athena-output).

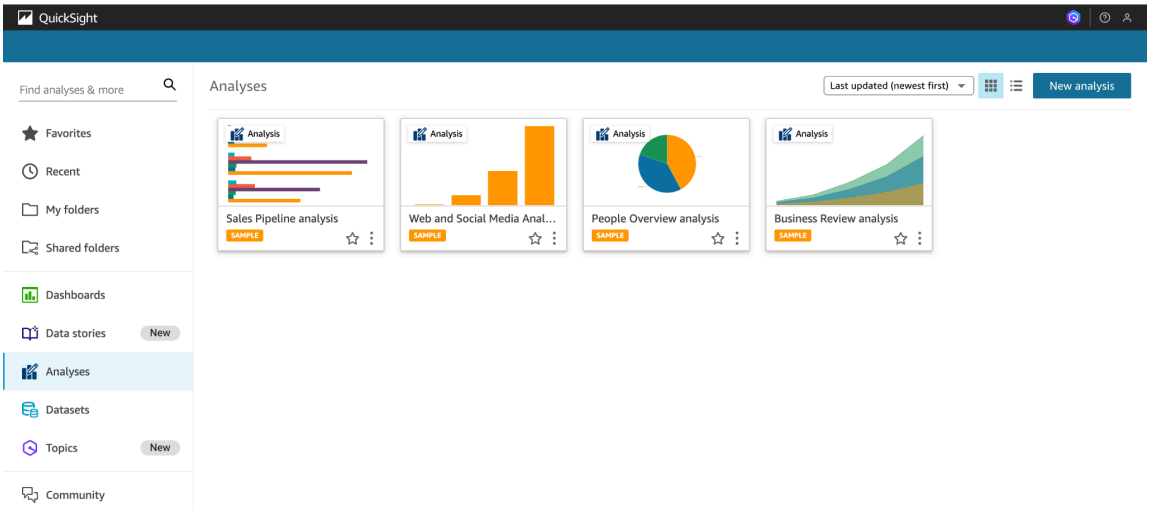


Step 6: Visualizing Data with AWS QuickSight

1. **Sign Up for AWS QuickSight:**

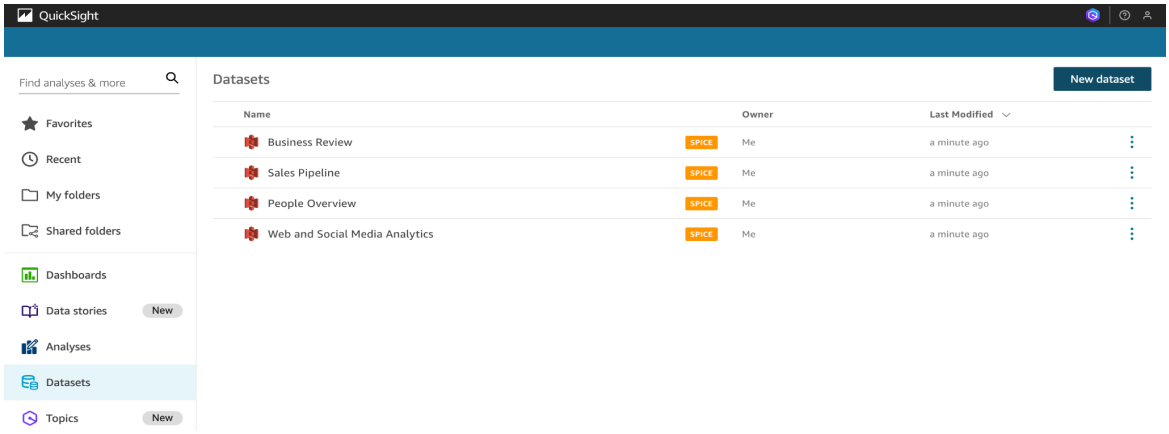
- Login to your root account and In the AWS Management Console, search for "QuickSight" and select the service.
- If you haven't signed up for QuickSight, you'll need to do so. Select the Enterprise Edition (free for 30 days).



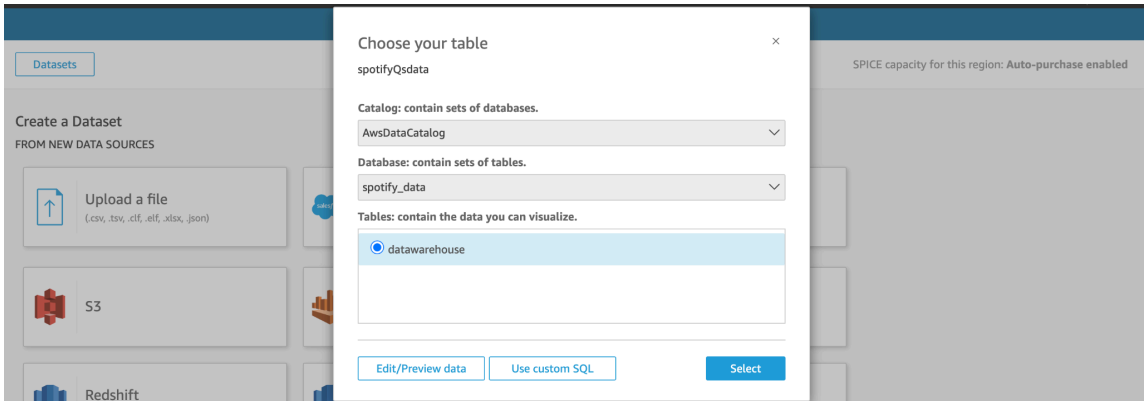


2. **Connect QuickSight to Athena:**

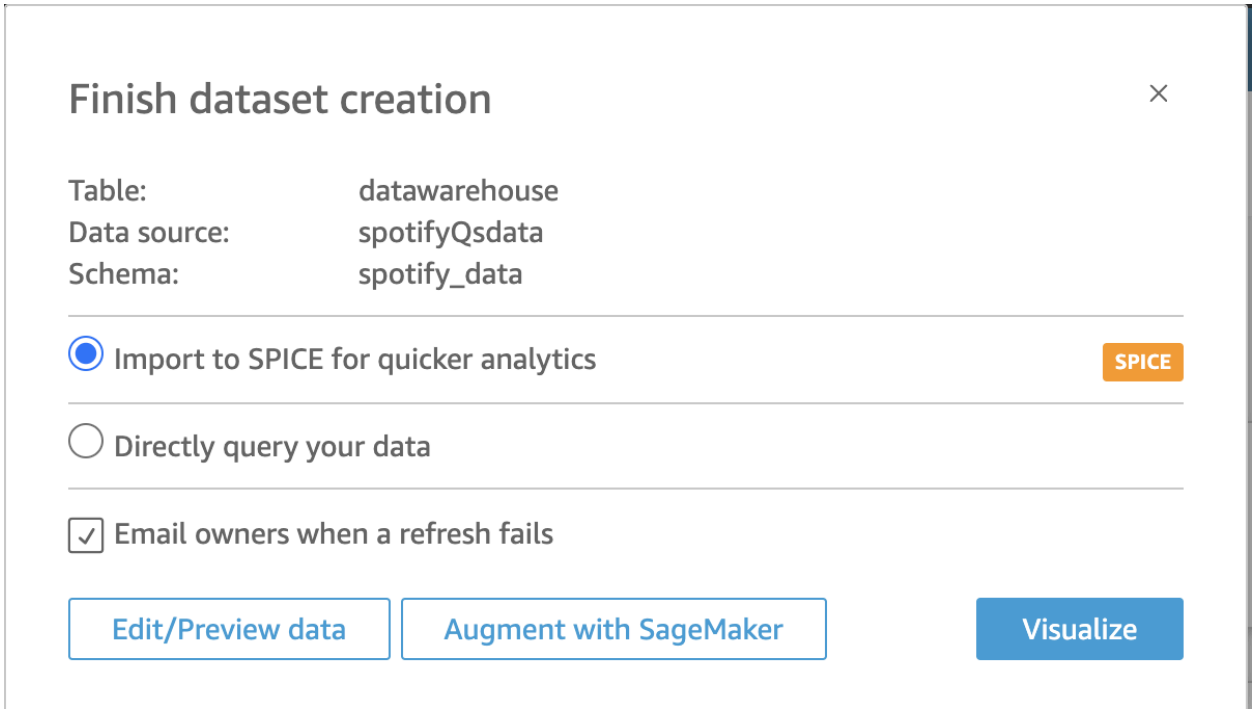
- Once signed in, go to "Datasets" and click "New dataset."



- Select "Athena" as the data source and name it with SpotifyQsData
- Choose the spotify_data database and the data_warehouse table.

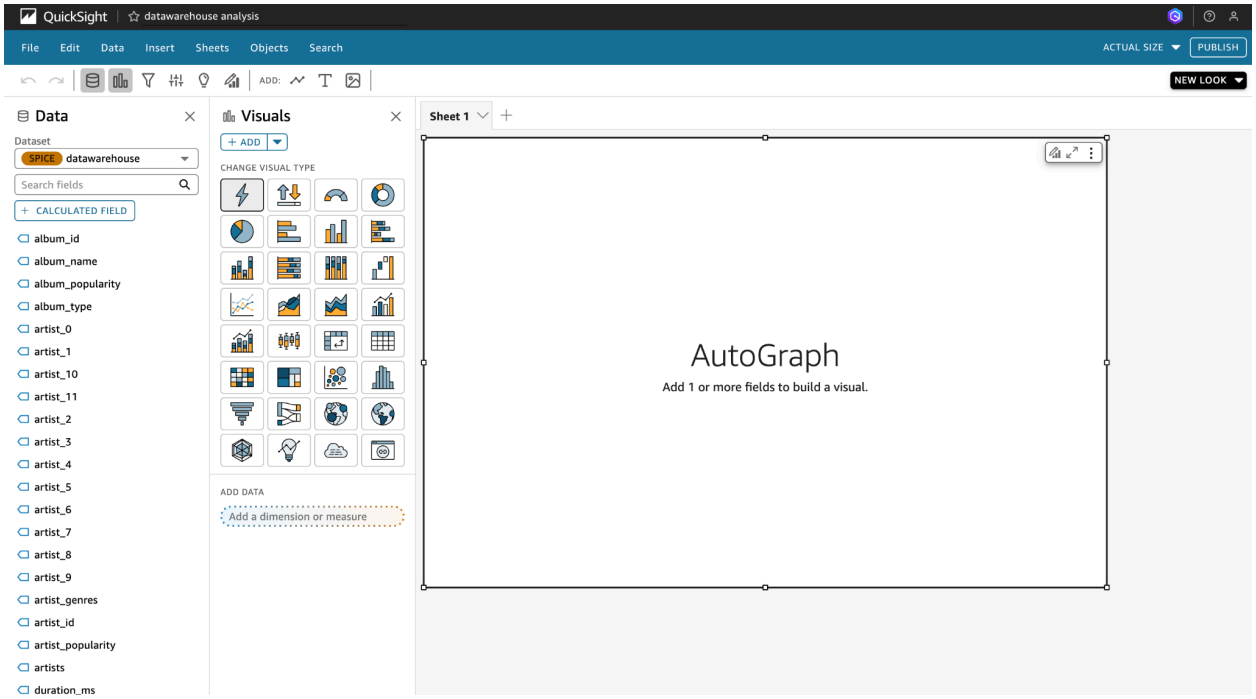


- Click on Visualize.



3. **Create Visualizations:**

- After importing the data, you can create various types of visualizations (e.g., bar charts, line charts, pie charts) using the fields from the data_warehouse table.



- Example: Create a bar chart to visualize the popularity of tracks by artist.

4. **Publish and Share Dashboards:**

- Once your visualizations are ready, you can publish the dashboard and share it with stakeholders.
- Click "Publish dashboard" and follow the prompts to share it via email or a link.

Conclusion

This document serves as a runbook for the Spotify Data Engineering Project. Follow each step to set up and run your end-to-end data pipeline.

Real-World Applications

The insights gained from this project can be applied in various real-world scenarios:

1. **Music Recommendations**: By analyzing the popularity of tracks and artists, platforms can improve their recommendation engines.
2. **Market Analysis**: Record labels and music producers can use the data to understand trends and make informed decisions on which genres or artists to promote.
3. **User Engagement**: Streaming services can analyze user preferences and behavior to enhance user engagement through personalized playlists and features.
4. **Business Intelligence**: The visualizations and insights derived can help business analysts make data-driven decisions to optimize marketing strategies and improve user retention.