Spotify End-End DE Project using AWS

Problem Statement:

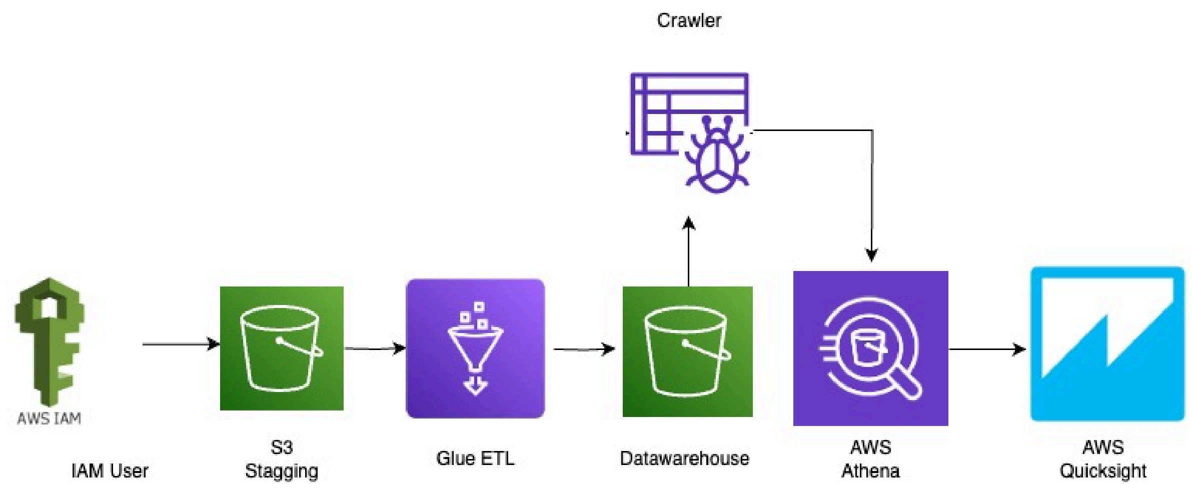
- In the rapidly evolving music streaming industry, data-driven insights are crucial for understanding user behavior, preferences, and trends.
- However, ingesting, processing, and analyzing massive volumes of data from platforms like Spotify can be a complex and resource-intensive endeavor.
- Traditional data engineering pipelines often require significant infrastructure setup, maintenance, and skilled personnel, leading to high operational costs and delayed time-to-insights.
- To address these challenges, there is a need for a scalable, serverless, and cost-effective data engineering solution that can streamline the process of ingesting, transforming, and analyzing data from Spotify.
- The solution should leverage cloud-native services to automate data ingestion, transformation, storage, and analysis, enabling data engineers and analysts to focus on extracting valuable insights rather than managing infrastructure.

Proposed Solution :

1. **Data Ingestion**: Provide a seamless and secure way to ingest raw data from Spotify, supporting various data formats and sources.
2. **Serverless Data Processing**: Implement a serverless extract, transform, and load (ETL) pipeline that can handle data transformations, data quality checks, and data enrichment without the need for provisioning and managing dedicated ETL servers or clusters.
3. **Scalable Data Storage**: Offer a scalable and cost-effective data storage solution that can accommodate growing data volumes from Spotify while ensuring high performance for queries and analytics.
4. **Metadata Management**: Automatically catalog and maintain metadata about the ingested data, including schemas, tables, and partitions, enabling efficient data discovery and querying.
5. **Interactive Data Analysis**: Provide an interactive and user-friendly interface for querying and analyzing the processed Spotify data using standard SQL, enabling data analysts and business users to extract insights without specialized skills or tools.
6. **Data Visualization and Reporting**: Offer powerful data visualization and reporting capabilities, allowing users to create interactive dashboards, reports, and visualizations to effectively communicate insights and facilitate data-driven decision-making.
7. **Security and Compliance**: Incorporate robust security measures and compliance features to protect sensitive data and ensure adherence to industry standards and regulations.

By implementing this serverless data engineering solution, organizations in the music streaming industry can unlock the full potential of their Spotify data, gain valuable insights into user behavior and preferences, and drive data-driven decision-making while minimizing infrastructure overhead and operational costs.

Architecture:



This architecture diagram illustrates a data engineering pipeline using various AWS services for **ingesting, processing, and analyzing** data from Spotify.

Spotify Data set used: https://www.kaggle.com/datasets/tonygordonjr/spotify-dataset-2023/data

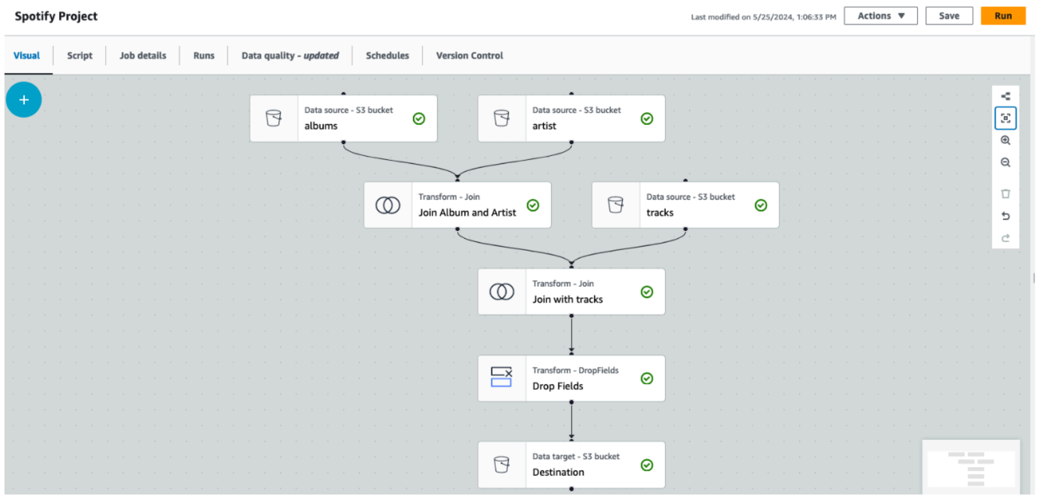
Here's an explanation of the components involved:

1. **AWS IAM (Identity and Access Management)**: The IAM User represents an authenticated user or application that initiates the data ingestion process.
2. **Amazon S3 (Simple Storage Service)**: S3 acts as a staging area where the raw data from Spotify is stored. This data could be in various formats, such as JSON, CSV, or Parquet.
3. **AWS Glue ETL**: AWS Glue is a fully managed extract, transform, and load (ETL) service. It crawls the data stored in the S3 staging area, infers the schema, and performs data transformations as defined in the ETL jobs. Glue ETL is responsible for cleaning, transforming, and preparing the data for analysis.
4. **Data Warehouse**: The transformed data from the Glue ETL process is loaded into a data warehouse, which could be Amazon Redshift, Amazon Athena, or a combination of both. The data warehouse serves as a centralized repository for storing and querying the processed data.
5. **Crawler**: The Crawler component, which is part of AWS Glue, periodically crawls the data in the data warehouse and updates the metadata in the AWS Glue Data Catalog. This metadata includes information about the data schemas, tables, and partitions, making it easier to discover and query the data.
6. **AWS Athena**: Athena is an interactive query service that allows you to analyze data directly from the data warehouse using standard SQL queries. It leverages the metadata stored in the AWS Glue Data Catalog to understand the structure of the data and perform queries.
7. **AWS QuickSight**: QuickSight is a cloud-native business intelligence (BI) service that allows you to create visualizations, dashboards, and reports based on the data queried from Athena or other data sources. It enables data analysts and business users to explore and analyze the Spotify data visually.

Summary:

1. The IAM User (or an application) uploads raw data from Spotify to Amazon S3 bucket.
2. AWS Glue ETL retrieves the data from S3, applies transformations, and loads the processed data into the data warehouse

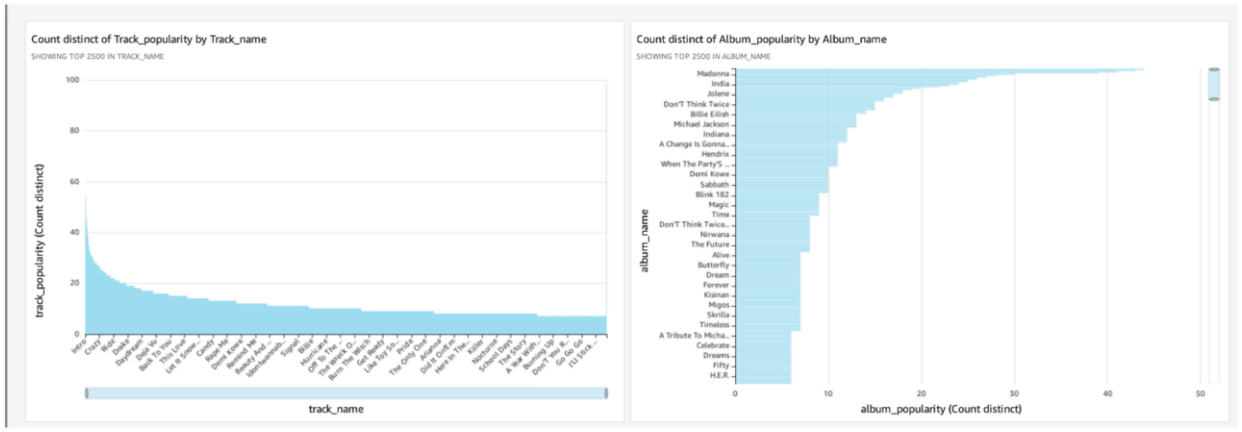
(e.g., Redshift or Athena).

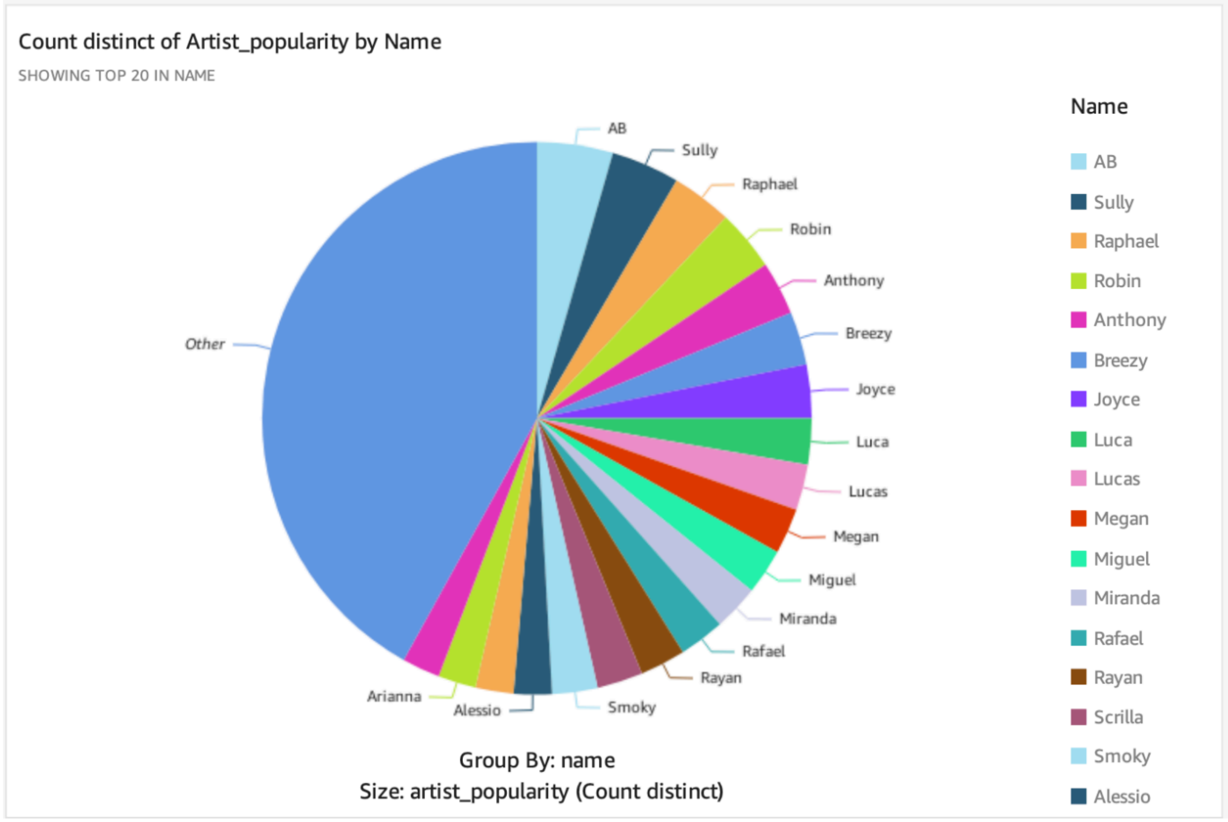
3. 

4. The Crawler component in AWS Glue crawls the data in the data warehouse and updates the metadata in the AWS Glue Data Catalog.
5. AWS Athena uses the metadata from the Data Catalog to query and analyze the data stored in the data warehouse.
6. The queried data from Athena can be visualized and analyzed further using AWS QuickSight, which creates dashboards and reports based on the Spotify data.

This architecture leverages serverless services like AWS Glue, Athena, and QuickSight, minimizing the need for managing infrastructure while providing a scalable and cost-effective solution for ingesting, processing, and analyzing Spotify data.

Quicksight:





1. Top Artists
2. Top Albums
3. Top Tracks

Appendix:

- Spotify Data set from a raw csv was manually cleaned and converted into 3 cleaned, structured csv's - Artist, Albums, Tracks.
- Created IAM role for AWS Glue to provide full access to S3.
- In real world scenario, data might be coming into S3 staging bucket via Databases, other sources, etc.
- Using Visual ETL, helps generate PySpark code using AWS Services.
- From Source in Visual ETL, select S3 for all 3 source files we have. Provide path, file format etc.
- Perform necessary Transformation by using Inner Join on Albums and Artists csv. Then join this with Track csv
- Drop all the unnecessary columns by joining the latest Album, Artist and Track CSV with Drop Fields element.
- Now use S3 as Destination to store this transformed data. Format of this data is Parquet. Compression Snappy
- Choose IAM Role for ETL Glue job to run. It generates the script in Pyspark.
- Now create DB in Glue, its like a Folder/Bucket. Then create Crawler to run on the DW S3 bucket to create Tables with metadata. Run the crawler.