Technical Report: Final Project DS 5110: Spotify Analytics Pipeline with AWS, Snowflake, and Power BI

Team Members: Ankur Haritosh, Pavan Kumar
Khoury College of Computer Sciences
Data Science Program

December 11, 2024

Contents

1 Introduction

This project focuses on building a fully automated serverless ETL pipeline to extract, transform, and visualize Spotify's playlist data. The objective is to automate the entire process of data extraction, transformation, and loading (ETL) into Snowflake for advanced analytics and insights using Power BI.

Automation is achieved using AWS CloudWatch triggers, Lambda functions, S3 for storage, and Snowpipe in Snowflake for seamless data ingestion. The pipeline generates actionable insights into song trends, artist performance, and release patterns.

2 Literature Review

Existing research highlights the benefits of automation in data pipelines for reducing manual intervention and ensuring scalability. However, there is limited work integrating serverless architectures with data warehouses like Snowflake. By combining AWS services, Snowflake, and Power BI, this project demonstrates an efficient and scalable solution for handling large datasets.

3 Methodology

The methodology involves three phases: automated data collection, transformation, and visualization, ensuring a fully serverless and scalable implementation.

3.1 Data Collection

Spotify API data was extracted every 8 hours using AWS Lambda functions triggered by CloudWatch. Raw JSON data was stored in an S3 bucket, ensuring real-time availability for downstream processing.

3.2 Data Preprocessing

S3 event triggers invoked a second Lambda function to normalize the data into three entities: 'Songs', 'Artists', and 'Albums'. The normalized data was stored as CSV files in S3. Snowpipe automatically ingested the transformed data into Snowflake tables, further automating the pipeline.

3.3 Analysis Techniques

SQL views in Snowflake were used to aggregate and organize data for analysis. Power BI dashboards provided interactive visualizations. Additionally, a machine learning model trained on Snowflake data was used to predict song popularity based on features like duration and artist attributes.

4 Results

Key results from the project include:

- Automation: All processes from data extraction to ingestion into Snowflake were automated using CloudWatch, event triggers, and Snowpipe.
- ETL Pipeline: Successfully extracted, transformed, and loaded data into Snowflake for analytical processing.
- Power BI Dashboards: Interactive insights into artist performance, song trends, and release timing.
- Machine Learning Model: Predicted song popularity, aiding playlist optimization.

5 Discussion

The project demonstrates the benefits of serverless architecture for building scalable, automated pipelines. Challenges included maintaining data quality and integrating machine learning into the ETL process. These were addressed through event-driven architectures and Snowflake's querying capabilities.

6 Conclusion

This project successfully implemented a serverless ETL pipeline integrating AWS, Snowflake, and Power BI for Spotify playlist analytics. The automated workflow reduced manual intervention, and the addition of a machine learning model enhanced playlist optimization. Future enhancements include broader playlist analysis and embedding predictive analytics directly into dashboards.

7 References

References

A Appendix A: Code

```python
# Spotify Data Transformation Lambda Function
def lambda_handler(event, context):
    playlist_links = ["https://open.spotify.com/playlist/37i9dQZF1DXcBWIGoYBM5M"]
    raw_data = extract_spotify_data(playlist_links)
    normalized_data = normalize_data(raw_data)
    upload_to_s3(normalized_data, 'spotify-etl-pipeline-sn/transform_data/')
```

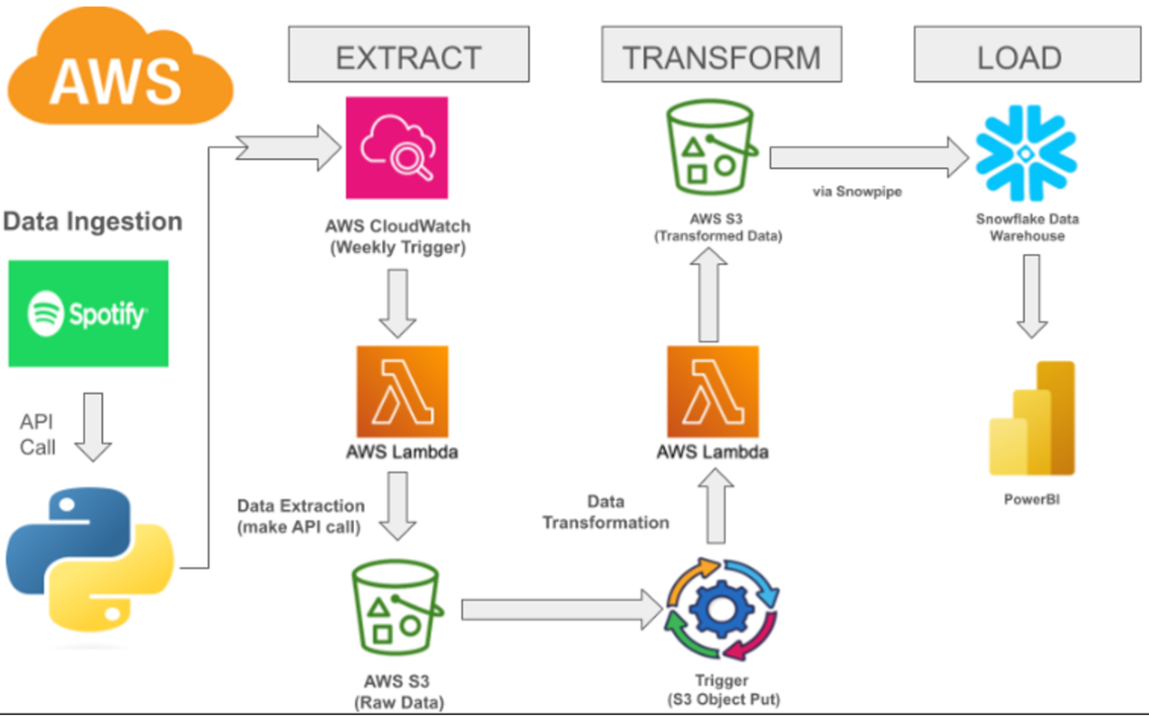
Listing 1: Lambda Function for Transformation

```sql
# Snowflake/SpotifyPlaylist_PowerBI_Views.sql

CREATE OR REPLACE VIEW SpotifyPlaylist_DW.Views.top_5_artists_view AS
WITH DistinctSongs AS (
    SELECT DISTINCT song_id, popularity, artist_id
    FROM SpotifyPlaylist_DW.SpotifyPlaylist_DW_SCHEMA.song
)
SELECT
    artist.artist_name,
    COUNT(song.song_id) AS song_count
FROM
    DistinctSongs AS song
JOIN
    SpotifyPlaylist_DW.SpotifyPlaylist_DW_SCHEMA.artist AS artist
    ON song.artist_id = artist.artist_id
WHERE
    song.song_id IN (
        SELECT song_id
        FROM DistinctSongs
        ORDER BY popularity DESC
        LIMIT 50
    )
GROUP BY
    artist.artist_name
ORDER BY
    song_count DESC
LIMIT 5;

SELECT * FROM SpotifyPlaylist_DW.Views.top_5_artists_view;
```

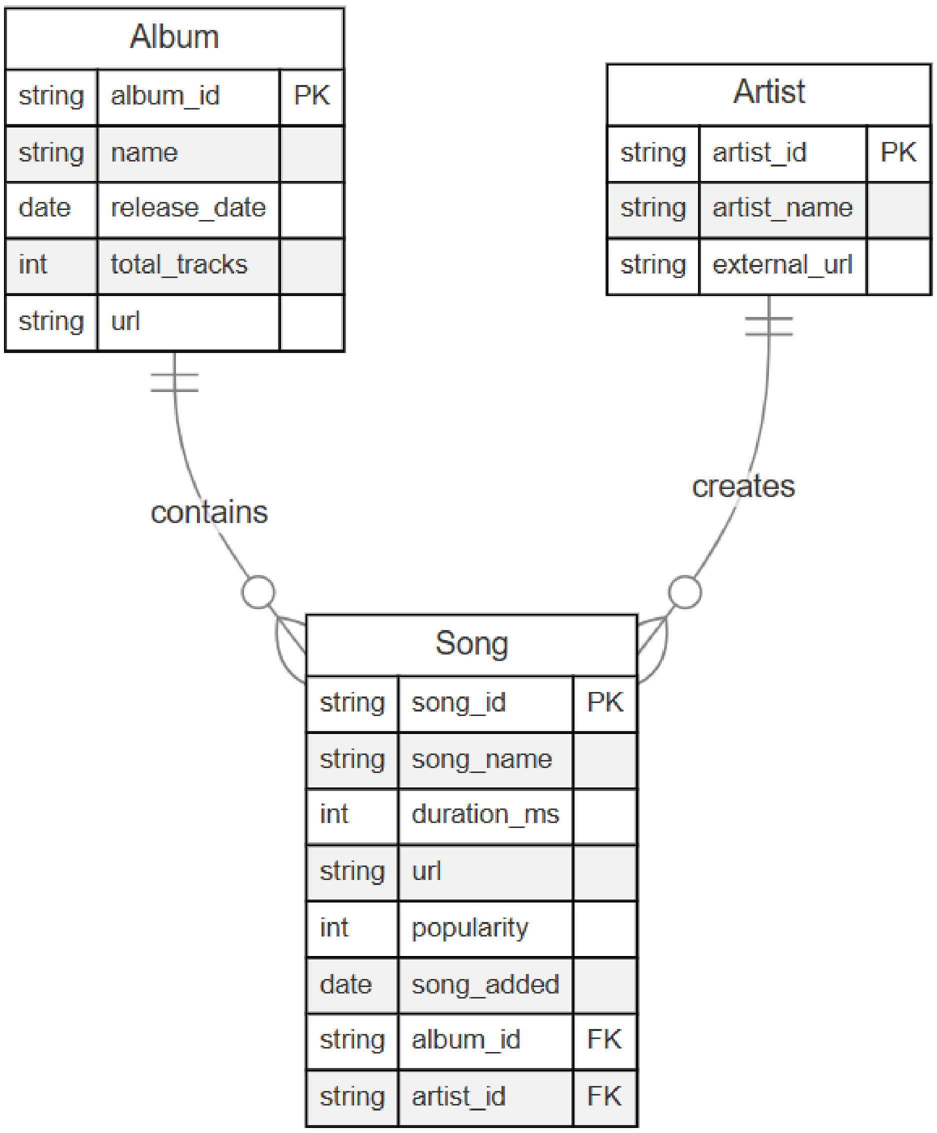
Listing 2: SpotifyPlaylist PowerBI SQL Views

B Appendix B: Additional Figures

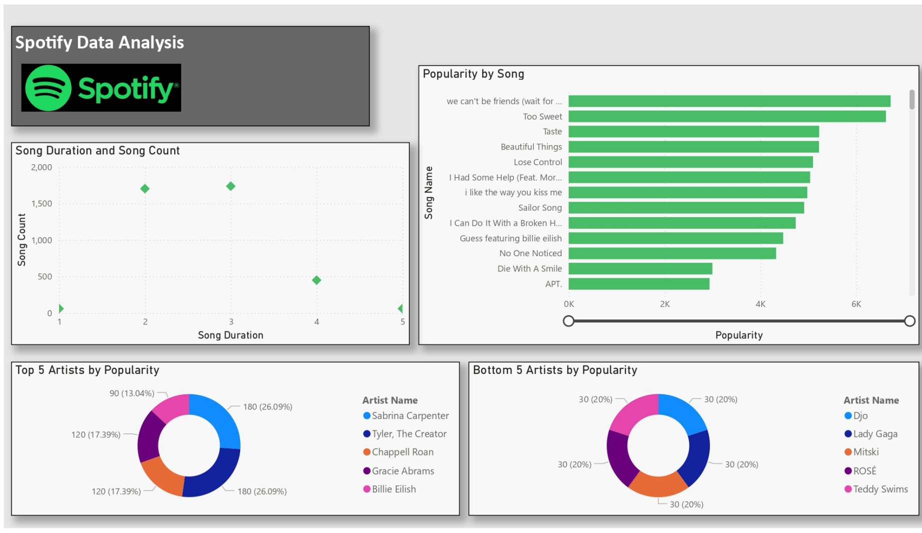
B.1 Pipeline Architecture



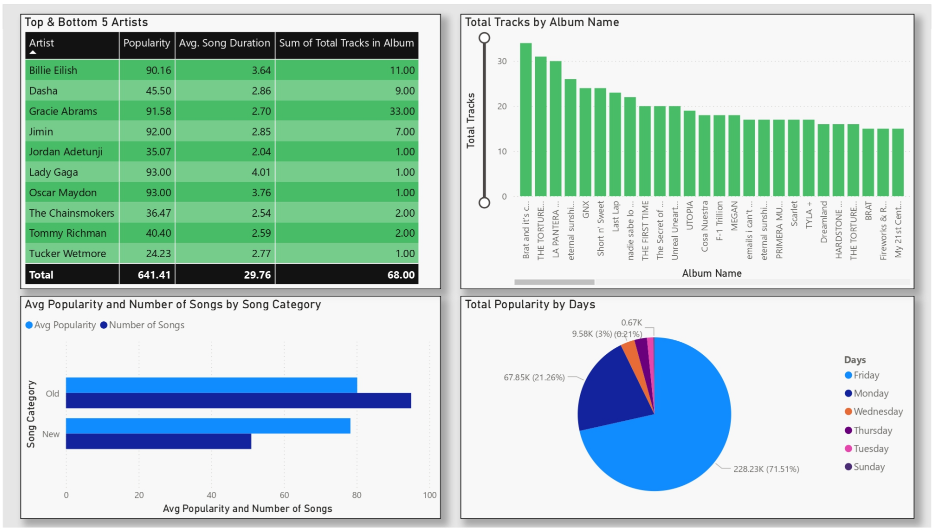
B.2 Data Model



B.3 Power BI Dashboards



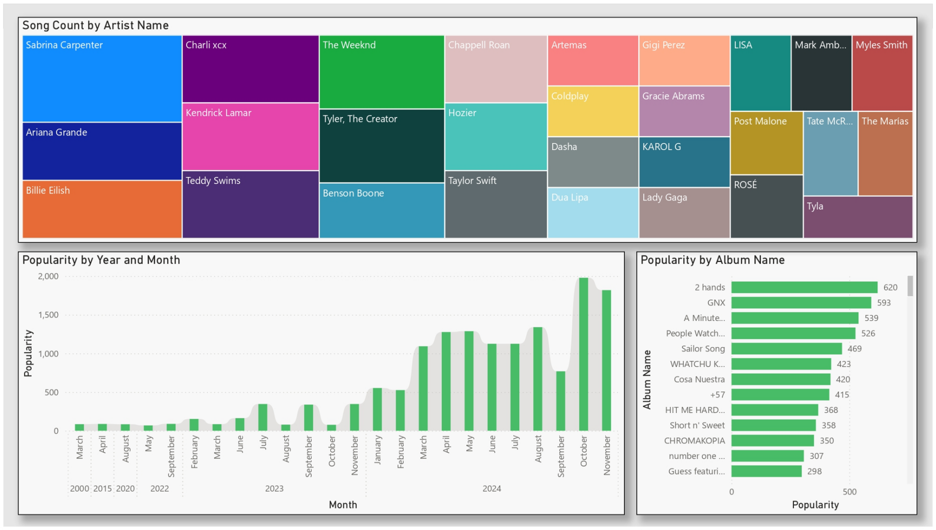
- Artist performance metrics:



Top & Bottom 5 Artists

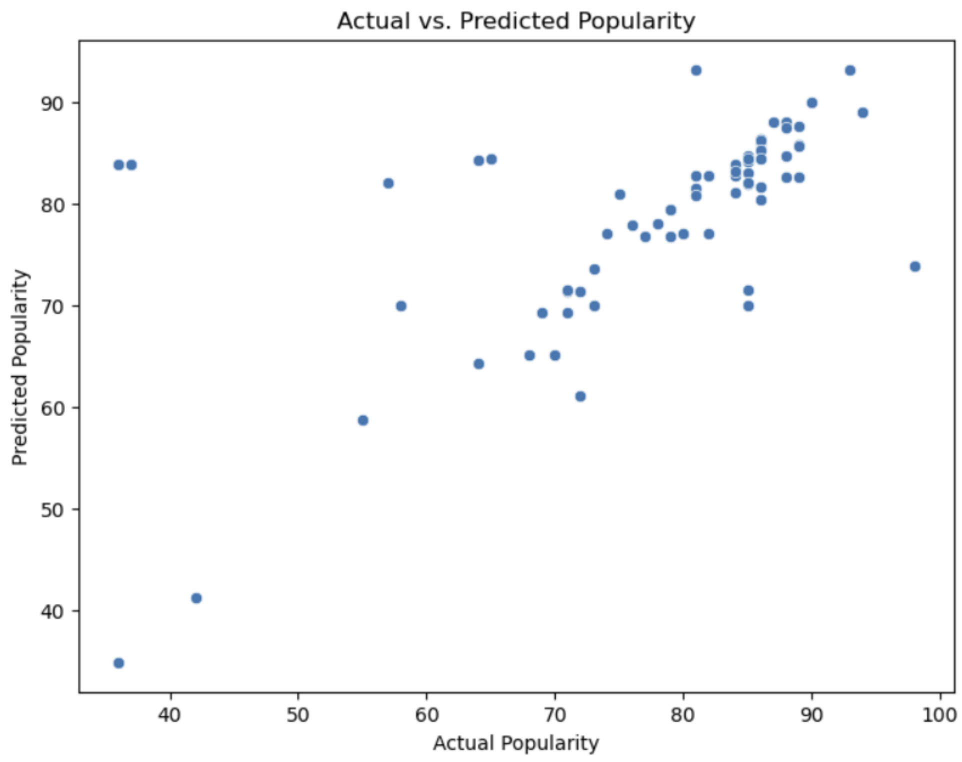
Artist	Popularity	Avg. Song Duration	Sum of Total Tracks in Album
Billie Eilish	90.16	3.64	11.00
Dasha	45.50	2.86	9.00
Gracie Abrams	91.58	2.70	33.00
Jimin	92.00	2.85	7.00
Jordan Adetunji	35.07	2.04	1.00
Lady Gaga	93.00	4.01	1.00
Oscar Maydon	93.00	3.76	1.00
The Chainsmokers	36.47	2.54	2.00
Tommy Richman	40.40	2.59	2.00
Tucker Wetmore	24.23	2.77	1.00
Total	**641.41**	**29.76**	**68.00**

- Song popularity trends:

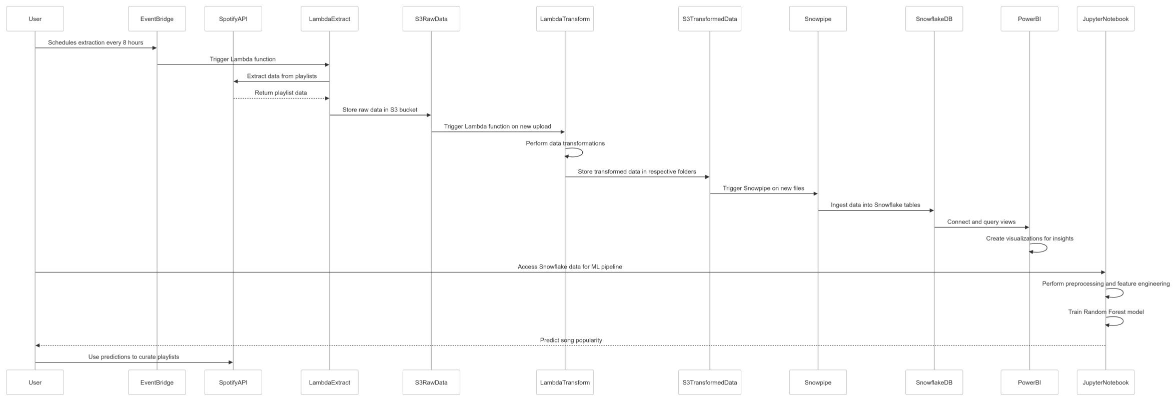


- Release timing analysis:

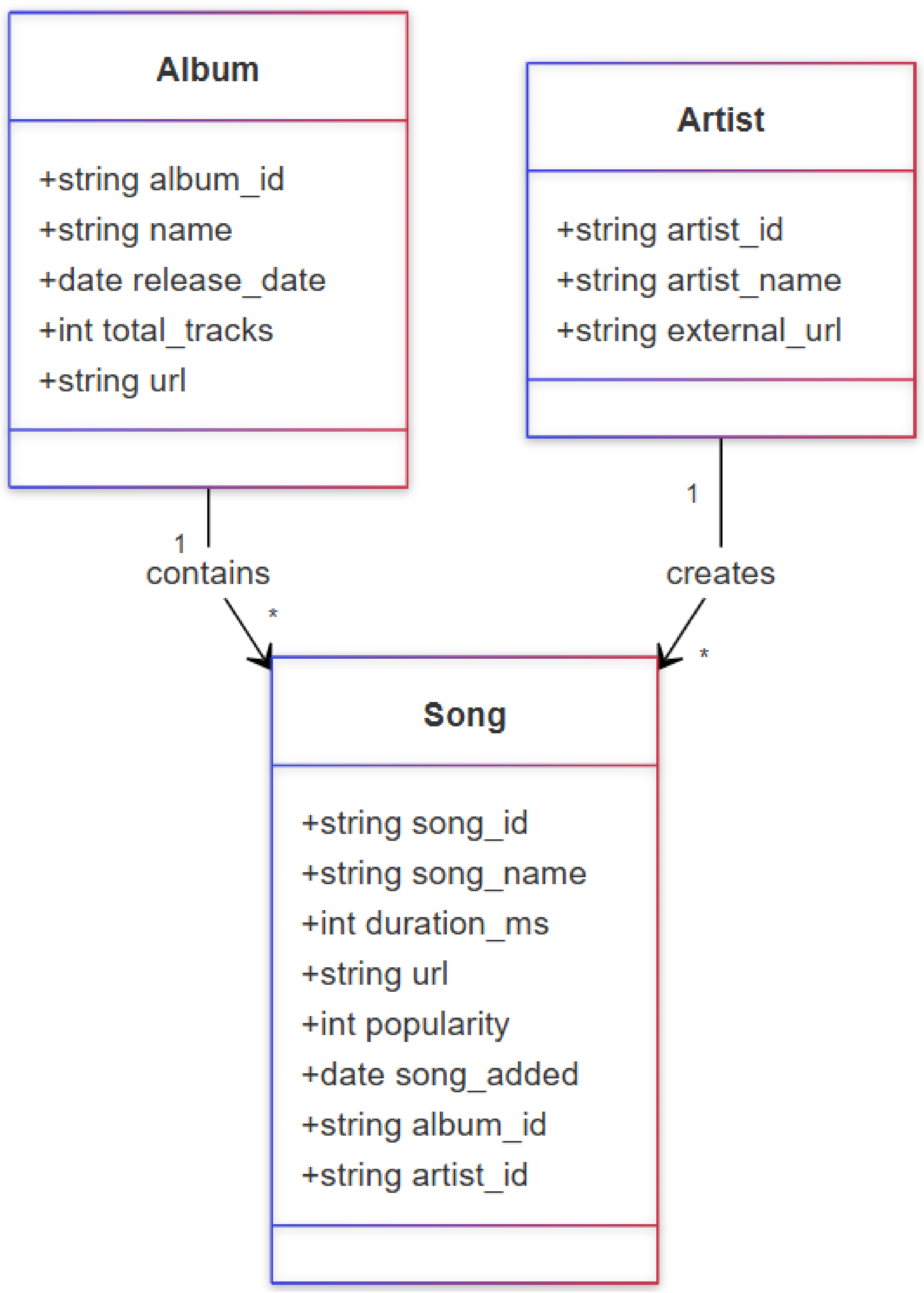
B.4 Machine Learning Model



B.5 Activity Diagram


User
EventBridge
SpotifyAPI
LambdaExtract
S3RawData
LambdaTransform
S3TransformedData
Snowpipe
SnowflakeDB
PowerBI
JupyterNotebook
Schedules extraction every 8 hours
Trigger Lambda function
Extract data from playlists
Return playlist data
Store raw data in S3 bucket
Trigger Lambda function on new upload
Perform data transformations
Store transformed data in respective folders
Trigger Snowpipe on new files
Ingest data into Snowflake tables
Connect and query views
Create visualizations for insights
Access Snowflake data for ML pipeline
Perform preprocessing and feature engineering
Train Random Forest model
Predict song popularity
Use predictions to curate playlists
User
EventBridge
SpotifyAPI
LambdaExtract
S3RawData
LambdaTransform
S3TransformedData
Snowpipe
SnowflakeDB
PowerBI
JupyterNotebook

B.6 Class Diagram


Album
+string album_id
+string name
+date release_date
+int total_tracks
+string url
Artist
+string artist_id
+string artist_name
+string external_url
1
contains
*
1
creates
*
Song
+string song_id
+string song_name
+int duration_ms
+string url
+int popularity
+date song_added
+string album_id
+string artist_id