Technical Report: Final Project DS 5110: Introduction to Data Management and Processing

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

November 18, 2024

Contents

1 Introduction

This project focuses on building a serverless ETL pipeline that extracts, transforms, and visualizes Spotify's "Top 100" playlist data. The objective is to automate data processing and create meaningful visual insights using Power BI. The pipeline integrates multiple AWS services, Snowflake, and Python libraries to handle data efficiently and generate analytical insights.

The scope is limited to playlist data from Spotify API, emphasizing song, artist, and album performance. This project excludes deep artist analysis and historical comparisons beyond the playlist data.

2 Literature Review

Existing research highlights the growing importance of automated data pipelines for business intelligence. Previous works focus on the use of APIs, ETL tools, and cloud-based solutions. However, there is limited work combining serverless architectures with Snowflake for analytical workflows. This project addresses this gap by implementing a scalable solution with advanced visualization.

3 Methodology

The methodology involves three primary phases: cloud setup, ETL pipeline implementation, and Power BI dashboard creation.

3.1 Data Collection

Data was collected using Spotify's API for the "Top 100" playlist. AWS Lambda functions automated weekly extraction and stored raw JSON data in an S3 bucket. API authentication was managed via environment variables.

3.2 Data Preprocessing

Raw JSON data was processed using a Lambda function triggered by S3 events. Data was normalized into three entities: Songs, Artists, and Albums. Transformed data was stored in CSV format in separate S3 directories.

3.3 Analysis Techniques

The transformed data was loaded into Snowflake using Snowpipe. Analytical SQL views were created for artist performance, song popularity, and duration trends. Power BI was used for interactive visualization.

4 Results

Key results include:

- Data Integration: Successful automation of data extraction, transformation, and loading into Snowflake.
- SQL Views: Metrics such as top artists by popularity, song duration distribution, and album track counts.
- Power BI Dashboard: Interactive visualizations for song trends, artist performance, and release timing.

Figures and tables illustrating these results are provided in Appendix B.

5 Discussion

The results demonstrate the efficiency of serverless architectures for automated data pipelines. Compared to traditional ETL workflows, this approach is scalable and reduces operational overhead. Challenges included learning advanced Power BI features, which were addressed through tutorials and iterative improvements.

6 Conclusion

This project successfully implemented a serverless ETL pipeline with AWS and Snowflake, delivering actionable insights through Power BI. Future work could expand to include multiple playlists and historical data analysis, enhancing the dashboard with predictive analytics.

7 References

References

A Appendix A: Code

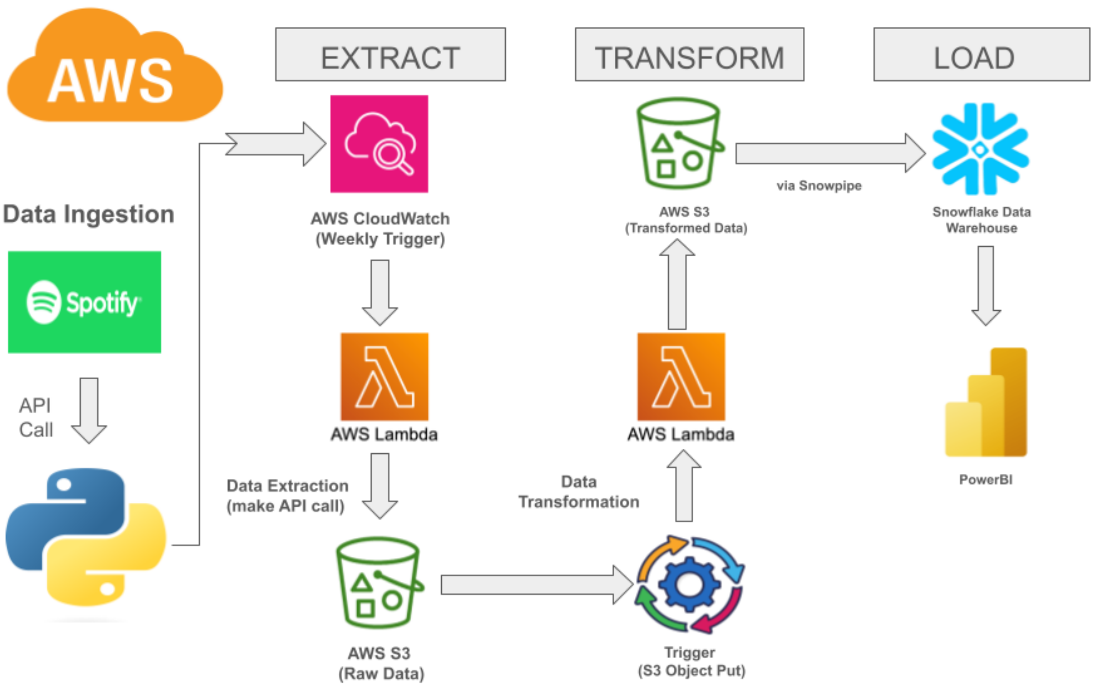
Below is an snippet used in data transformation:

```

client_credentials_manager = SpotifyClientCredentials(client_id=
    client_id, client_secret=client_secret)
sp = spotipy.Spotify(client_credentials_manager=
    client_credentials_manager)
client = boto3.client('s3')

def lambda_handler(event, context):
     playlist_links = ["https://open.spotify.com/playlist/37
    i9dQZF1DXcBWIGoYBM5M", "https://open.spotify.com/playlist/37
    i9dQZF1DX0ieekvzt1Ic",
          "https://open.spotify.com/playlist/37i9dQZF1DWY4lFlS4Pnso", "
    https://open.spotify.com/playlist/37i9dQZF1DX0kbJZpiYdZl"]
     playlist_names = ["TopHits", "HotHitsIndia", "HotHitsUK", "
    HotHitsUSA"]

    #Dictionary where the key is a playlist name and the value is the playlist_data
    spotify_data = {}

    for i in range(len(playlist_links)):
        playlist_URI = playlist_links[i].split('/')[-1]
        playlist_data = sp.playlist_tracks(playlist_URI)
        spotify_data[playlist_names[i]] = playlist_data

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

    #This is the Extract part -> Lambda pushes the unstructured data to S3
    client.put_object(
        Bucket="spotify-snowflake-etl-pavan",
        Key="raw_data/to_process/" + filename,
        Body=json.dumps(spotify_data)
    )
```

Listing 1: Lambda Function for Transformation

B Appendix B: Additional Figures

- A diagram of the ETL pipeline architecture.


AWS
Data Ingestion
Spotify
API Call
EXTRACT
AWS CloudWatch (Weekly Trigger)
AWS Lambda
Data Extraction (make API call)
AWS S3 (Raw Data)
TRANSFORM
AWS S3 (Transformed Data)
AWS Lambda
Data Transformation
Trigger (S3 Object Put)
via Snowpipe
LOAD
Snowflake Data Warehouse
PowerBI