MUSIC STREAMING PROJECT

A music streaming service requires an end-to-end data pipeline to analyze user streaming behavior.

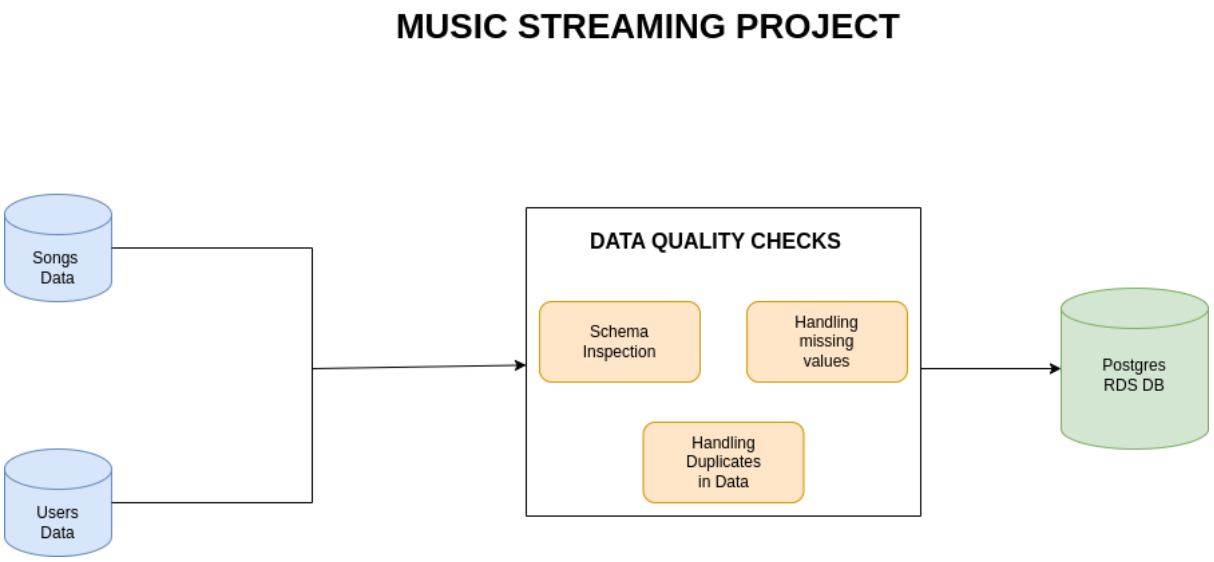
This pipeline aims to integrate data from multiple sources (AWS RDS and S3), process it, and generate key performance indicators (KPIs) for business intelligence.

Key Performance Indicators (KPIs):

- **Genre-Level KPIs:** Metrics that provide insights into how different music genres perform on the platform.
    1. **Listen Count:** Total number of times tracks in a genre have been played.
    2. **Average Track Duration:** The mean duration of all tracks within a genre.
    3. **Popularity Index:** A computed score based on play counts, likes, and
    4. shares for tracks in a genre.
    5. **Most Popular Track per Genre:** The track with the highest engagement
    6. (plays, likes, or shares) in each genre.

- **Hourly KPIs:** Metrics that capture platform activity trends on an hourly basis.
    1. **Unique Listeners:** The distinct number of users streaming music in a given hour.
    2. **Top Artists per Hour:** The most streamed artists during each hour.
    3. **Track Diversity Index:** A measure of how varied the tracks played in an hour are, based on the number of unique tracks played compared to total plays.

FLOW DIAGRAM



TABLES

Users: This table consists of information on users for the music streaming service.

Column Name	Column Type	Column Comment
User Name	Text	The name of the user streaming music from the music streaming service
User Age	Integer	Age of the user streaming the music
User Country	Text	The country the user is streaming the music from.
Created at	Text	The time the user created the account for streaming

Songs: This table consists of information about the songs being streamed.

Column Name	Column Type	Column Comment
Track id	Text	ID referencing the track
Artists	Text	The name of the artist.
Album Name	Text	Name of Album
Track Name	Text	Name of Track(Song)
Popularity	Integer	The level of popularity of the song(Track).
Duration ms	Integer	Duration of the track in minutes
Explicit	Boolean	PG contents in a track
Danceability	Real	The rate at which the songs(Tracks) can be danced to.
Energy	Real	The rate of energy in a track
Key	Integer	The musical key the track was written, sung and played in.
Loudness	Real	The rate of how high or low a track is
Mode	Integer	A musical scale coupled with a set of characteristics, melodious and harmonic behaviors.
Speechiness	Real	The rate of words present in the track
Acousticness	Real	The rate of no-electrical

		instruments in the track
Instrumentalness	Real	The rate of vocals(voices) present in a track
Liveness	Real	The rate of the audience present while performing the track.
Valence	Real	The rate of emotional quality of the track
Tempo	Real	How slow or fast the track is
Time Signature	Integer	The structure of the beats and musical notes in a track.
Track Genre	Text	The music category the track belongs to

Streams: This table consists of information on the streaming from users.

Column Name	**Column Type**	**Column Comment**
User ID	Integer	ID referencing the user
Track ID	Text	ID referencing the track
Listen Time	Timestamp	The time the user started listening/streaming the music

ORCHESTRATION

Overview

The pipeline extracts data from an RDS database and Amazon S3, transforms it to compute key performance indicators (KPIs), and loads the results into Amazon Redshift for analytics and reporting. The pipeline is designed to handle large datasets efficiently by fetching data in batches and archiving processed data.

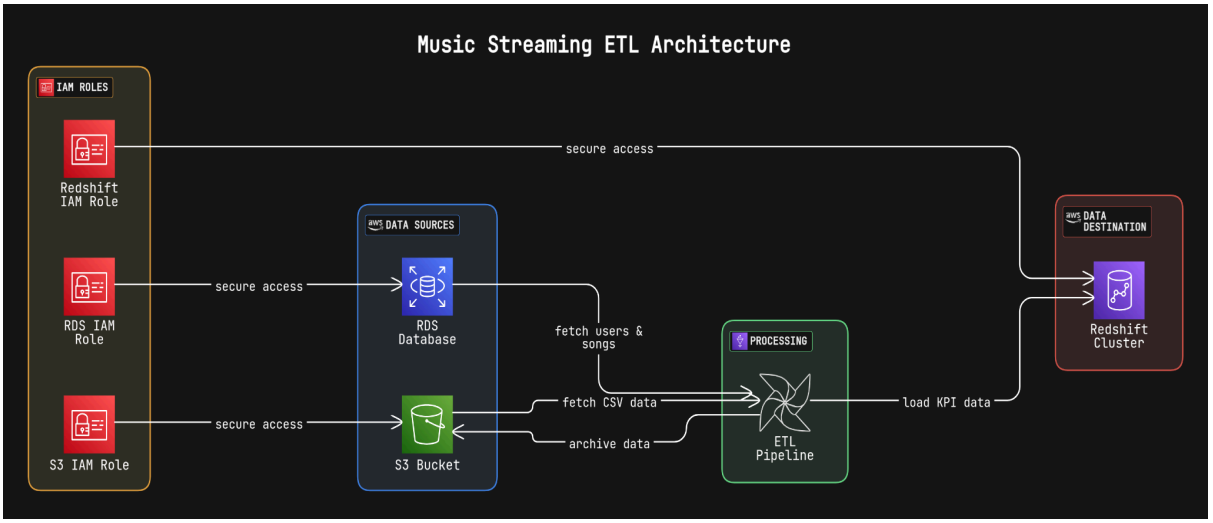
Pipeline Objectives

1. Extract:
    - Fetch user and song data from an RDS database in batches.
    - Fetch streaming data from an S3 bucket.
2. Transform:
    - Merge the extracted data into a single dataset.
    - Compute daily genre-level and hourly KPIs.
3. Load:
    - Load the computed KPIs into Amazon Redshift for further analysis.
4. Archive:
    - Archive processed S3 files and mark RDS data as processed.

Pipeline Architecture

The pipeline consists of the following components:

1. RDS Database: Stores user and song data.
2. Amazon S3: Stores streaming data in CSV files.
3. Amazon Redshift: Stores computed KPIs for analytics.
4. Apache Airflow: Orchestrates the ETL workflow.



Pipeline Workflow

The pipeline is implemented as a Directed Acyclic Graph (DAG) in Airflow. Below is a breakdown of the tasks and their dependencies:

Tasks

1. **Check RDS Data:**
    - Checks if there is data in the users and songs tables in the RDS database.
    - If no data is found, the pipeline ends.
    - If data is found, proceed to fetch data.

2. **Fetch RDS Users:**
    - Fetches user data from the users table in batches.
    - Pushes the fetched data to XCom for use in downstream tasks.
3. **Fetch RDS Songs:**
    - Fetches song data from the songs table in batches.
    - Pushes the fetched data to XCom for use in downstream tasks.
4. **Check S3 Files:**
    - Checks if there are files in the specified S3 bucket.
    - If no files are found, the pipeline ends.
    - If files are found, proceed to fetch data.
5. **Fetch S3 Data:**
    - Fetches streaming data from the S3 bucket.
    - Pushes the fetched data to XCom for use in downstream tasks.
6. **Merge Data:**
    - Merges the user, song, and streaming data into a single DataFrame.

- Ensures the required columns (user_id, track_id) exist before merging.
- Pushes the merged data to XCom for use in downstream tasks.

7. **Compute KPIs:**
    - Computes daily genre-level and hourly KPIs from the merged data.
    - Pushes the computed KPIs to XCom for use in downstream tasks.
8. **Create KPI Table in Redshift:**
    - Creates a table named music_kpi_metrics in Redshift if it doesn't already exist.
9. **Load Genre KPIs:**
    - Loads the computed genre-level KPIs into the music_kpi_metrics table in Redshift.
10. **Load Hourly KPIs:**

    Loads the computed hourly KPIs into the music_kpi_metrics table in Redshift.

11. **Archive Data:**
    - Archives processed S3 files by moving them to an archive folder.
12. **End DAG:**
    - A dummy task to mark the end of the pipeline.

Key Features

1. **Batch Processing:**
    - Fetches data from RDS in batches to handle large datasets efficiently.

2. **Error Handling:**
    - Each task includes error handling to log and raise exceptions if something goes wrong.

3. **Dynamic Workflow:**
    - Based on data availability, it uses the BranchPythonOperator to decide whether to proceed or end the pipeline.

4. **Data Archiving:**
    - Archives processed S3 files to avoid reprocessing.

5. **Scalability:**
    - Designed to scale with increasing data volumes by leveraging batch processing and distributed computing (Redshift).

KPIs Computed

1. Daily Genre-Level KPIs
    - Listen Count: Total number of times tracks in a genre have been played.
    - Average Track Duration: Mean duration of all tracks within a genre.
    - Popularity Index: A computed score based on play counts, likes, and shares for tracks in a genre.
    - Most Popular Track per Genre: The track with the highest engagement (plays, likes, or shares) in each genre.

2. Hourly KPIs
    - Unique Listeners: The distinct number of users streaming music in a given hour.
    - Top Artists per Hour: The most streamed artists during each hour.
    - Track Diversity Index: A measure of how varied the tracks played in an hour are, based on the number of unique tracks played compared to total plays.

Redshift Table Schema

Column Name	Data Type	Description
track_genre	VARCHAR(255)	Genre of the track.
listen_count	INT	Total number of listens for the genre.
Avg_track_duration_sec	FLOAT	Average duration of tracks in the genre(in seconds)
popularity_index	FLOAT	Computed popularity score for the genre.
most_popular_track	VARCHAR(255)	Most popular track in the genre.
stream_hour	INT	Hour of the day (0-23).
unique_listeners	INT	Number of unique listeners

		in the hour.
top_artists	VARCHAR(255)	Most streamed artist in the hour.
track_diversity_index	FLOAT	Diversity index for tracks played in the hour.
kpi_type	VARCHAR(50)	Type of KPI (genre or hourly).

Technical Details

Dependencies

- Apache Airflow: Orchestrates the pipeline.
- Amazon RDS: Stores user and song data.
- Amazon S3: Stores streaming data.
- Amazon Redshift: Stores computed KPIs.
- Pandas: Used for data manipulation and transformation.
- Psycopg2: Used for interacting with PostgreSQL/Redshift.

Connections

- RDS Connection:
- Connection ID: aws_rds_connection
- Host: RDS endpoint
- Database: Name of the database.

- Username: RDS username.
- Password: RDS password.
- Port: 5432 (default for PostgreSQL).

S3 Connection:

- Connection ID: aws_conn
- AWS Access Key ID: Your AWS access key.
- AWS Secret Access Key: Your AWS secret key.

Redshift Connection:

- Connection ID: redshift_conn
- Host: Redshift endpoint
- Database: Name of the Redshift database.
- Username: Redshift username.
- Password: Redshift password.
- Port: 5439 (default for Redshift).

Pipeline Execution

The pipeline is scheduled to run daily (schedule_interval='@daily'). It can be triggered manually or automatically based on the schedule.

Monitoring and Logging

- Airflow UI: Provides real-time monitoring of task status, logs, and retries.
- Logging: Each task logs its progress and errors for easy debugging.

Future Enhancements

Incremental Data Processing:

- Process only new or updated data to improve efficiency.

Data Validation:

- Add data validation steps to ensure data quality.

Alerting:

- Integrate with alerting tools (e.g., Slack, PagerDuty) to notify stakeholders of pipeline failures.

Dashboard:

- Build a dashboard (e.g., using Tableau, Power BI, or Amazon QuickSight) to visualize the computed KPIs.

Conclusion

The Music Streaming ETL Pipeline is a robust and scalable solution for extracting, transforming, and loading music streaming data. It provides valuable insights into user behavior and platform performance, enabling data-driven decision-making. The pipeline is designed to handle large datasets efficiently and can be extended to support additional use cases in the future.