Batch Data Processing of Music Streams Using Airflow & Redshift

# Executive Summary

This document outlines the batch data processing workflow implemented using Apache Airflow, Amazon Redshift, and AWS S3. The purpose of this project is to process music streaming data, perform transformations, and generate key insights for reporting and analytics.

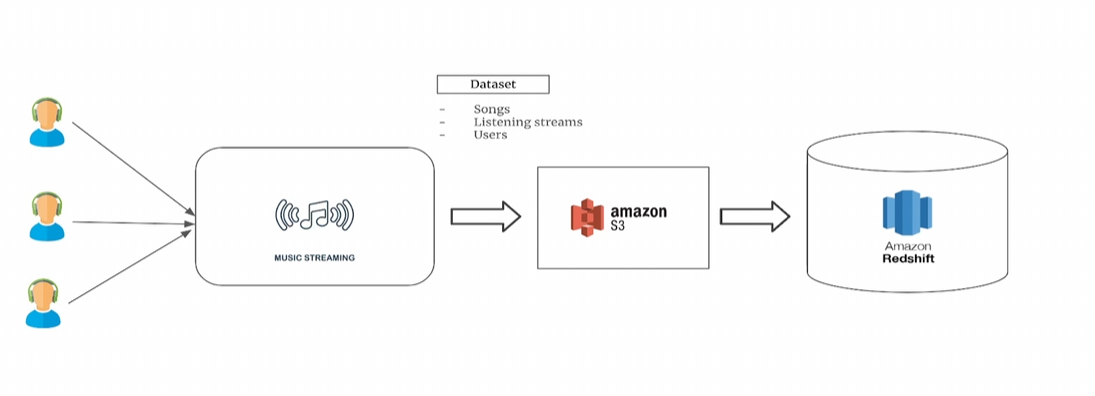
# Introduction

The project involves processing music streaming data using a batch ingestion pipeline. Music streaming data is frequently uploaded to an AWS S3 bucket and must be processed to extract useful analytics. Apache Airflow is used to orchestrate the data ingestion, validation, transformation, and loading into Amazon Redshift for reporting.

# Solution Architecture

The solution consists of three main layers:  
1. \*\*Storage Layer\*\*: Data is stored in Amazon S3 in CSV format.  
2. \*\*Orchestration Layer\*\*: Apache Airflow manages the ingestion, validation, and transformation.  
3. \*\*Data Warehouse\*\*: Amazon Redshift stores the processed data for analytics.

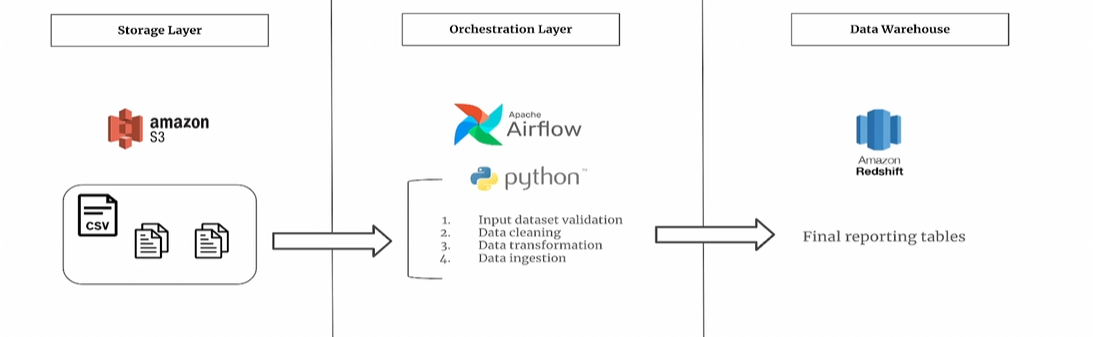
The high-level architecture is depicted below:



# Data Pipeline Workflow

The batch data pipeline is managed through an Airflow DAG that consists of the following steps:  
1. \*\*Data Ingestion\*\*: Input datasets (songs, users, and streams) are read from S3.  
2. \*\*Validation\*\*: The DAG verifies the required columns are present.  
3. \*\*Transformation\*\*: KPIs such as genre-level insights and hourly engagement metrics are calculated.  
4. \*\*Loading\*\*: The transformed data is ingested into Redshift.  
5. \*\*Archiving\*\*: Processed files are moved to an archive folder.

The detailed workflow is shown in the following diagram:



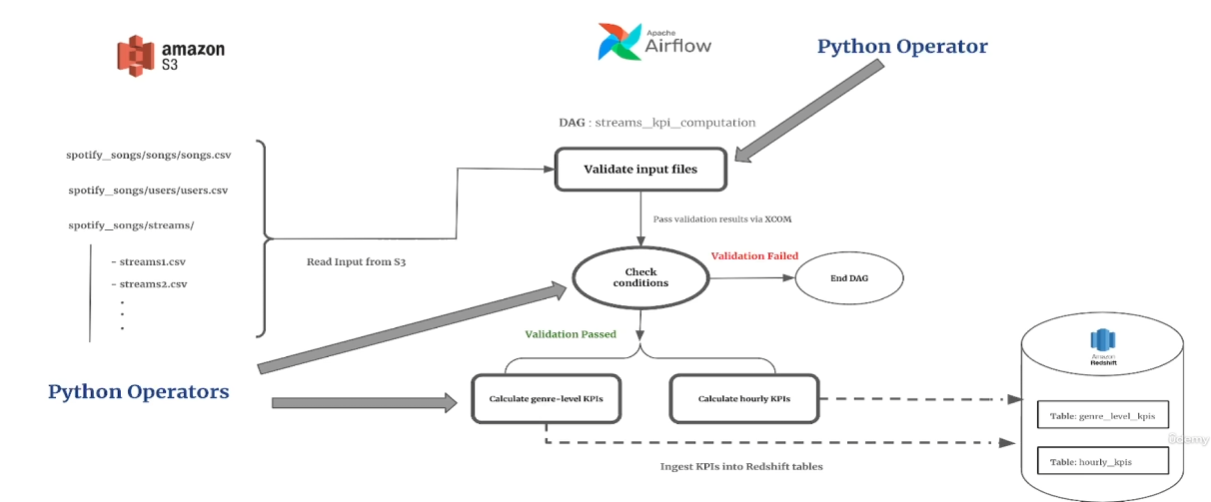
# Technologies Used

- \*\*Amazon S3\*\*: Acts as the data lake for storing raw and processed data.  
- \*\*Apache Airflow\*\*: Orchestrates the data pipeline execution.  
- \*\*Python & Pandas\*\*: Used for data transformation.  
- \*\*Amazon Redshift\*\*: Data warehouse for analytical reporting.

# Implementation Details

The Airflow DAG is implemented with Python operators and hooks to interact with AWS services. Validation results are shared between tasks using XComs, and data transformations are handled using Pandas. The Redshift database schema consists of two primary tables: `genre\_level\_kpis` and `hourly\_kpis`. The data is ingested using an UPSERT strategy to ensure freshness.

The Airflow DAG execution logic is as follows:



# Conclusion & Recommendations

This project successfully demonstrates the use of AWS services for large-scale data processing. The combination of S3, Airflow, and Redshift allows for efficient batch processing and analysis of music streaming data. Future enhancements could include:  
- Implementing incremental data loading for efficiency.  
- Utilizing Airflow sensors for real-time event-driven execution.  
- Expanding the Redshift schema to include additional analytics use cases.