Music Streaming Services Project Documentation

Project Overview

This data pipeline is designed to process real-time streaming data from a music streaming service. The goal is to validate, transform, and compute key performance indicators (KPIs) for business insights, using Apache Airflow for orchestration and AWS Glue for data transformation. The processed data is stored in Amazon DynamoDB for fast retrieval by downstream applications.

Architecture Overview

The pipeline follows these key stages:

1. **Data Ingestion:**
    - Streaming data arrives in Amazon S3 under the streams/ folder.
    - Additional data such as songs and users is also ingested from their respective folders.
2. **Data Validation:**
    - The DAG checks and validates required columns in the datasets.
    - If the required columns are not included in the datasets, the DAG ends.
3. **Data Transformation:**
    - Streaming, songs, and user data are combined and saved as temporary files for further processing.
4. **Data Cleaning & KPI Calculation:**
    - AWS Glue is triggered to compute KPIs such as:
        - Daily Genre-Level KPIs (listen count, unique listeners, total listening time, average listening time per user, top 3 songs per genre per day, top 5 genres per day).
5. **Data Storage:**
    - The transformed data is stored in DynamoDB for fast lookups.
6. **File Archival:**
    - Successfully processed streams are moved to an s3 glacier bucket for archival purposes.

Airflow DAG Overview

DAG Structure

- **DAG Name:** music_streaming_pipeline
- **Schedule Interval:** @daily
- **Start Date:** 2025-03-21

Tasks & Flow

1. **start_dag_task**
    - Initial task that marks the start of the DAG.
2. **sense_streaming_data** (BranchPythonOperator)
    - Continuously checks the streams/ folder for streams and proceeds with the subsequent tasks if the bucket receives a file.
3. **extract_and_combine_streams_task** (PythonOperator)
    - Combines streaming, songs, and user data into temporary files.
    - Pushes file paths to XCom for downstream tasks.
4. **validate_csv_files_task** (BranchPythonOperator)
    - Validates the structure of CSV files.
    - Branches to either:
        - move_files_to_intermediate_bucket_task (if validation passes)
        - end_dag_if_validation_fails_task (if validation fails)
5. **move_files_to_intermediate_bucket_task** (PythonOperator)
    - Moves combined and validated files to an intermediate bucket.
6. **trigger_glue_job** (GlueJobOperator)
    - Triggers an AWS Glue job which transforms and computes KPI's.
7. **delete_files_from_intermediate_bucket_task** & **archive_streams_data_task** (PythonOperators)
    - Ensures processed files are cleared from the intermediate bucket and streams are archived for tracking purposes.
8. **end_dag_task** (EmptyOperator)
    - Marks the end of the DAG.

Key Features

- **Robust Validation:** Ensures all required columns exist before processing.
- **Efficient Transformation:** Uses AWS Glue for PySpark transformations to compute KPIs efficiently.
- **Resilient Error Handling:** Detailed logging and defined failure paths enhance reliability.
- **Scalable Design:** Optimized for handling unpredictable data arrivals and large data volumes.

Instructions to Run the Pipeline

1. Deploy the DAG in Apache Airflow using MWAA (Managed Workflows for Apache Airflow).
2. Ensure AWS credentials are properly configured with the required permissions.
3. Set up AWS Glue jobs for data transformation and DynamoDB insertion. The script for the transformation is included in the project directory.
4. Trigger the DAG manually or let it run on schedule.

NB: The airflow dag is located in the dag folder in the project directory.

Sample DynamoDB Queries

- Retrieve top 5 genres for a given date:

```
SELECT genre, listen_count FROM music_kpis_table

WHERE date = '2025-03-10'

ORDER BY listen_count DESC

LIMIT 5;
```

- Retrieve top 3 songs per genre for a given date:

```
SELECT genre, song, listen_count FROM music_kpis_table

WHERE date = '2025-03-10'

ORDER BY genre, listen_count DESC

LIMIT 3;
```

Troubleshooting

- **Missing Files:** Ensure the data files are uploaded to the correct S3 paths.

- **Glue Job Errors:** Review AWS Glue job logs for PySpark errors or schema mismatches.
- **XCom Errors:** Ensure xcom_push() keys are correctly referenced in downstream tasks.

For further details, refer to the code implementation and comments in the DAG file.