ETL Pipeline for Music Streaming Data

Overview

This project implements an **ETL (Extract, Transform, Load)** pipeline using **Apache Airflow** to analyze user streaming behavior for a music streaming service. The pipeline integrates data from multiple sources, processes it, and generates **Key Performance Indicators (KPIs)** for business intelligence. The processed data is loaded into **Amazon Redshift** for further analysis.

Architecture

The system architecture consists of the following components:

1. **Data Sources**:
   - **Amazon RDS**: Stores user and song metadata (simulated using CSV datasets).
   - **Amazon S3**: Stores streaming data in batch files.
2. **ETL Pipeline**:
   - **Extract**: Data is extracted from Amazon RDS (PostgreSQL) and Amazon S3.
   - **Transform**: The data is validated, transformed, and KPIs are computed.
   - **Load**: The processed data is loaded into Amazon Redshift.
3. **Orchestration**:
   - **Apache Airflow**: Orchestrates the entire ETL pipeline, ensuring tasks are executed in the correct order.
4. **Destination**:
   - **Amazon Redshift**: Stores the processed KPIs for analytical queries.

ETL Pipeline Workflow

1. Extract

- **Streaming Data**:
  - The pipeline checks for streaming data files in an S3 bucket (streaming-data-nsp24).

    - If files are found, they are combined into a single CSV file (combined_streaming_data.csv).
- **User and Song Metadata**:
    - Data is extracted from an Amazon RDS PostgreSQL database using SQL queries.
    - The extracted data is saved as CSV files (users_data.csv and songs_data.csv).

2. Validate

- The pipeline validates that all required columns are present in the extracted data:
    - **Streaming Data**: user_id, track_id, listen_time.
    - **User Data**: user_id, user_name, user_age, user_country, created_at.
    - **Song Data**: track_id, artists, album_name, track_name, popularity, duration_ms, etc.

3. Transform

- The streaming data is merged with song metadata to compute the following KPIs:
    - **Genre-Level KPIs**:
        - Listen Count
        - Average Track Duration
        - Popularity Index
        - Most Popular Track per Genre
    - **Hourly KPIs**:
        - Unique Listeners
        - Top Artists per Hour
        - Track Diversity Index
- The computed KPIs are saved as CSV files (genre_kpis_daily.csv and hourly_kpis_daily.csv).

4. Load

- The KPI CSV files are uploaded to an S3 bucket (kpis-intermediate-bucket).
- The data is loaded into Amazon Redshift using the COPY command.

Tasks in the DAG

1. **validate_streams_in_s3**:
   - Validates if there are files in the S3 bucket.
   - Branches to extract_and_combine_streams if files exist, otherwise ends the DAG.
2. **extract_and_combine_streams**:
   - Extracts and combines streaming data files from S3 into a single CSV file.
3. **extract_users_and_songs_from_postgres**:
   - Extracts user and song metadata from Amazon RDS and saves them as CSV files.
4. **validate_columns**:
   - Validates that all required columns are present in the extracted data.
   - Branches to transform_and_compute_kpis if validation passes, otherwise ends the DAG.
5. **transform_and_compute_kpis**:
   - Computes genre-level and hourly KPIs and saves them as CSV files.
6. **create_redshift_tables**:
   - Creates the genre_kpis and hourly_kpis tables in Amazon Redshift if they don’t already exist.
7. **upload_csv_to_s3**:
   - Uploads the KPI CSV files to an S3 bucket.
8. **load_data_into_redshift**:
   - Loads the KPI data from S3 into Amazon Redshift using the COPY command.

Task Dependencies

The tasks are executed in the following order:

1. **validate_streams_in_s3** → **extract_and_combine_streams** or **end_dag_if_no_streams_exists_task**.

2. **extract_and_combine_streams** → **validate_columns**.
3. **extract_users_and_songs_from_postgres** → **validate_columns**.
4. **validate_columns** → **transform_and_compute_kpis** or **end_dag_if_columns_missing_task**.
5. **transform_and_compute_kpis** → **create_redshift_tables** → **upload_csv_to_s3** → **load_data_into_redshift**.

Setup Instructions

1. Prerequisites

- **Amazon RDS**: A PostgreSQL database with users and songs tables.
- **Amazon S3**: A bucket (streaming-data-nsp24) for streaming data and another bucket (kpis-intermediate-bucket) for KPI data.
- **Amazon Redshift**: A Redshift cluster with the necessary IAM role for accessing S3.
- **Apache Airflow**: Installed and configured with the required connections (aws_default, aws_postgres_conn, aws_redshift_conn).

2. Airflow Connections

- **aws_default**: AWS connection for accessing S3.
- **aws_postgres_conn**: PostgreSQL connection for Amazon RDS.
- **aws_redshift_conn**: Redshift connection for Amazon Redshift.

3. Running the DAG

- Deploy the DAG (music_streaming_etl.py) to your Airflow environment.
- Trigger the DAG manually or schedule it to run daily.

Troubleshooting

Common Issues

1. **S3 Bucket Not Found**:
   - Ensure the bucket names (streaming-data-nsp24 and kpis-intermediate-bucket) are correct and accessible.
2. **Missing Columns**:

  - Verify that the extracted data contains all required columns.

3. **Redshift Connection Issues**:
   - Ensure the Redshift cluster is running and the IAM role has the necessary permissions.
4. **CSV File Errors**:
   - Check that the CSV files are properly formatted and do not contain invalid data.

Conclusion

This ETL pipeline provides a robust solution for analyzing music streaming data. By leveraging Apache Airflow, Amazon RDS, Amazon S3, and Amazon Redshift, the pipeline ensures efficient data processing and storage for business intelligence purposes.