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.

SQL Queries for Validating KPIs in Redshift

These queries ensure that the data has been correctly computed and loaded into the genre_kpis and hourly_kpis tables.

1. Genre-Level KPIs

a. Total Listen Count per Genre

```
SELECT
  track_genre,
  SUM(listen_count) AS total_listen_count
FROM
  genre_kpis
GROUP BY
  track_genre
ORDER BY
  total_listen_count DESC;
```

b. Average Track Duration per Genre

```
SELECT
  track_genre,
  AVG(avg_duration_ms) AS avg_track_duration_ms
FROM
  genre_kpis
GROUP BY
  track_genre
ORDER BY
  avg_track_duration_ms DESC;
```

c. Popularity Index per Genre

```
SELECT
  track_genre,
  AVG(popularity_index) AS avg_popularity_index
FROM
  genre_kpis
GROUP BY
  track_genre
```

```
ORDER BY
  avg_popularity_index DESC;
```

d. Most Popular Track per Genre

```
SELECT
  date,
  track_genre,
  most_popular_track,
  most_popular_track_popularity
FROM
  genre_kpis
ORDER BY
  date, track_genre;
```

2. Hourly KPIs

a. Unique Listeners per Hour

```
SELECT
  date,
  hour,
  SUM(unique_listeners) AS total_unique_listeners
FROM
  hourly_kpis
GROUP BY
  date, hour
ORDER BY
  date, hour;
```

b. Top Artists per Hour

```
SELECT
  date,
  hour,
```

```
    top_artist
FROM
    hourly_kpis
ORDER BY
    date, hour;
```

c. Track Diversity Index per Hour

```
SELECT
    date,
    hour,
    AVG(track_diversity_index) AS avg_track_diversity_index
FROM
    hourly_kpis
GROUP BY
    date, hour
ORDER BY
    date, hour;
```

How to Use These Queries

1. **Connect to Redshift:** Use a SQL client (e.g., Amazon Redshift Query Editor, DBeaver, or psql) to connect to your Redshift cluster.
2. **Run the Queries:** Execute the queries to validate the KPIs stored in the genre_kpis and hourly_kpis tables.

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.