**TV Viewership ETL Pipeline Design Document**

**Glossary**

| **Term** | **Definition** |
| --- | --- |
| ETL | Extract, Transform, Load - A process of collecting, cleaning, and storing data. |
| Airflow | An open-source orchestration tool for scheduling and monitoring workflows. |
| Redshift | Amazon's data warehouse service for large-scale data storage and querying. |
| Raw Data Table | A table that stores unprocessed or minimally processed data for auditing. |
| Downstream Table | A table optimized for analytics, containing minute-by-minute viewership data. |
| Program Metadata | Supplemental data containing program details like name, genre, and ID. |
| Partitioning | A method to divide data into subsets for improved processing efficiency. |
| Dynamic Partition Pruning | A Spark feature to limit scanned data by filtering relevant partitions dynamically. |
| Broadcast Join | A method to optimize joins by sending smaller datasets to all worker nodes. |

**1. Introduction**

The TV Viewership ETL Pipeline processes and analyzes logs generated by set-top boxes to derive insights into viewer behavior and channel popularity. The system ingests raw log data in JSON format, enriches it with program metadata, and stores the processed data in Amazon Redshift for analysis. The pipeline is automated using Apache Airflow to ensure daily execution.

This document outlines the pipeline’s design, including the tools and technologies used, data flow, Redshift table schemas, and scalability considerations.

**2. Data Pipeline Overview**

**2.1 Pipeline Objectives**

* Process raw log data and program metadata daily.
* Clean, validate, and enrich the raw data.
* Store processed data in Redshift for analytics and reporting.
* Automate the entire pipeline using Airflow.

**2.2 High-Level Architecture**

1. **Data Ingestion**:
   * Source: JSON log files and program data in text format from the AWS s3(local filesystem is used in current case).
   * Tool: Apache Spark reads and validates the data.
2. **Data Transformation**:
   * Enrichment: Join raw logs with program metadata.
   * Deduplication: Remove duplicate records.
   * Aggregation: Calculate minute-level viewership and user durations.
3. **Data Loading**:
   * Destination: Amazon Redshift with two tables:
     + Raw Data Table
     + Downstream Table
4. **Automation**:
   * Orchestrated by Airflow, which schedules and monitors the ETL pipeline daily.

**3. Redshift Table Schemas**

**3.1 Raw Data Table**

Stores minimally processed data for auditing and troubleshooting purposes.

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| mac\_id | VARCHAR | Unique identifier for the set-top box. |
| datetime | TIMESTAMP | Combined event date and time. |
| channel\_name | VARCHAR | Name of the channel being viewed. |
| program\_id | VARCHAR | Unique ID of the program. |
| geo\_location | VARCHAR | Geographical location of the user. |
| event\_code | VARCHAR | Type of event (e.g., channel change). |
| satellite\_name | VARCHAR | Name of the satellite. |
| timestamp | VARCHAR | Raw event timestamp. |
| index\_time | VARCHAR | Time when the event was indexed. |
| duration\_in\_seconds | INTEGER | Duration of the viewing session. |

**3.2 Downstream Table**

Stores processed data optimized for analytics, retaining the latest channel viewed within the same minute.

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| mac\_id | VARCHAR | Unique identifier for the set-top box. |
| datetime | TIMESTAMP | Minute-level timestamp of the event. |
| channel\_name | VARCHAR | Name of the channel being viewed. |
| program\_id | VARCHAR | Unique ID of the program. |

**4. Tools and Technologies**

| **Tool** | **Purpose** |
| --- | --- |
| **Apache Spark** | Distributed data processing and transformation. |
| **Amazon Redshift** | Data warehouse for storing processed datasets and analytics for Business analytics in AWS environment. |
| **Apache Airflow** | Orchestrates the ETL pipeline for daily automation. |
| **Python** | Used to implement the ETL logic and Airflow DAG scripts. |
| **SQL** | Used as a Querying language in Redshift workspace. |

**Amazon S3** Used as a data lake to store raw data from the source.

**5. Scalability Considerations**

**5.1 Handling Large-Scale Data**

1. **Partitioning**:
   * Data is partitioned by datetime during transformations to reduce scan times and improve query performance.
2. **Dynamic Partition Pruning**:
   * Spark ensures only relevant partitions are scanned, optimizing large-scale data processing.
3. **Broadcast Joins**:
   * Used to efficiently join smaller metadata tables with larger datasets.

**5.2 Optimizing Redshift**

1. **Sort Keys and Distribution Keys**:
   * Sort by datetime and distribute by program for efficient query execution.
2. **Materialized Views**:
   * Precompute frequently queried metrics (e.g., top channels) to reduce query latency.

**5.3 Future Enhancements**

* **Scalability considerations:**
* For extremely large datasets, data can be stored in partitioned Parquet files in Amazon S3 and queried via Redshift Spectrum, reducing storage costs and enhancing performance.
* Configure Spark and Redshift clusters to scale dynamically based on workload requirements, ensuring efficient resource utilization and cost management.
* **Cloud-Native Processing**:
  + Transition to Amazon S3 for storing raw and processed data, enabling integration with Redshift Spectrum for querying large datasets directly from S3.
  + Add more metadata sources (e.g., additional program attributes) to enrich the data further and support advanced analytics.
* **Real-Time Processing**:
  + Extend the pipeline to process logs in real-time using Amazon Kinesis or Kafka.
* **Error Handling and Monitoring**:
  + Integrate logging and alerting mechanisms in Airflow to track job status and failures.
  + Using CICD to deploy the codes in higher environments to minimize human errors

**6. Summary**

This ETL pipeline efficiently processes TV viewership logs and program metadata, storing enriched data in Redshift for analytics. Using Spark for distributed data processing and Airflow for automation ensures scalability, reliability, and ease of maintenance. The system is designed to handle increasing data volumes and supports future enhancements like real-time processing and cloud-native storage.