MySQL to Snowflake Data Pipeline Documentation

Overview

This document details the end-to-end automated data pipeline which extracts data from a MySQL database, transforms and uploads it to AWS S3, and loads the data into a Snowflake data warehouse using Apache Airflow.

Pipeline Architecture

The pipeline consists of 5 primary steps executed as Airflow tasks:
1. Export MySQL data to Parquet format.
2. Transform the exported data for consistency.
3. Upload transformed Parquet files to S3.
4. Load Parquet files from S3 into Snowflake using COPY INTO.
5. Archive the uploaded files in a timestamped ZIP on S3.

Airflow DAG

The DAG pipeline_mysql_dag is scheduled to run every 5 minutes and ensures tasks run in sequence:
1.export_mysql_to_parquet
2.transform_parquet_data_mysql
3.upload_to_s3_mysql
4.load_to_warehouse_mysql
5.archive_mysql_files_in_s3

Each task is implemented as a modular PythonOperator with logging both to Airflow and local files.

Task Details

1. Export MySQL to Parquet:
- Extracts review data using CDC logic.
- Applies hashing and generates inserts/updates/deletes.
- Writes data in chunks to data/mysql_exports_cdc.

2. Transform Parquet Data:
- Cleans nulls and inconsistent text.
- Deduplicates rows based on UserId, ProductId, and ProfileName.

- Converts timestamps and saves transformed files.

3. Upload to S3:
- Uploads only new transformed Parquet files to an S3 bucket.
- Skips files that already exist.

4. Load to Snowflake:
- Executes a COPY INTO statement from the external S3 stage.
- Uses Snowflake credentials and file format configured via .env.

5. Archive Files:
- Archives the uploaded Parquet files into ZIP format.
- Deletes source files post archive.
- Maintains clean state and prevents duplicates.

Data Integrity

The pipeline ensures data integrity via:
- Using review_ingest_tracker to track last max ID and avoid re-processing.
- Row hash comparison for change detection.
- A shadow file to detect deletions weekly.
- Strict ordering and atomic operations in Airflow DAG with max_active_runs=1 and locking mechanisms to avoid race conditions.

5. Performance Optimization

The pipeline is optimized for scalability and performance:
- Processes data in chunks (10,000 rows).
- Uses lightweight Parquet format for storage and upload.
- Skips unchanged rows via hashing.
- Weekly full deletion scan to avoid daily overhead.
- Dual logging to Airflow and custom logs folder for debugging.

6. Assumptions Made

- Id is a monotonically increasing primary key in the MySQL table.
- Data inflow order is based on Id.
- created_at is used as timestamp and should be a valid datetime string.
- Only one active DAG run is allowed to maintain order and avoid concurrency issues.

Logging Strategy

- Each task logs to a local logs/ folder and to Airflow UI.
- Separate log files are maintained for each Airflow task (e.g., export_mysql_to_parquet_airflow.log).

Configuration and Secrets

- Secrets are managed via a .env file in the project root.
- AWS, MySQL, and Snowflake credentials are securely loaded using dotenv.
- Parquet output directories and S3 prefixes are configurable.

Shadow File and Deletion Tracking

- Shadow file stores row hashes to detect inserts, updates, and deletes.
- Deletes are logged in an append-only Parquet log (data/deletion_history).

Recovery & Restart

- The review_ingest_tracker table tracks the last processed ID.
- Each task is idempotent and can be retried without data corruption.