Data Pipeline Service

(pipeline.py)

This service implements an ETL (Extract, Transform, Load) pipeline for moving and transforming data from various sources into a BigQuery data warehouse. The main entry point is the handler function, which processes a list of tables defined in **DATA_TABLES**. Below is an overview of the service's workflow and the purpose of its imports.

Imports and Their Functionalities

- **time**
  Used for measuring durations of extraction, transformation, and loading steps.
- **src.dealer**
  - **CUSTOM_TRANSFORMATIONS:** Dictionary mapping table names to their specific transformation functions.
  - **bigquery_init:** Initializes a BigQuery client.
  - **now_now_str:** Returns the current timestamp as a string.
  - **get_last_successful_runs:** Retrieves metadata about the last successful pipeline run for a table.
  - **fetch_incremental_data:** Fetches new/updated records since the last run.
  - **log_pipeline_run:** Logs metadata about each pipeline run.
  - **data_volume:** Calculates the size of the data in bytes and megabytes.
  - **iso_tz_to_epoch:** Converts ISO timestamp strings to epoch time.
  - **pipe_job:** Credentials or configuration for pipeline jobs.
  - **loader:** Loads data into BigQuery.
  - **google_worksspace_message:** Sends notifications (e.g., to Google Chat/Workspace).
  - **create_pipeline_summary:** Generates a summary message for each pipeline run.
- **src.utils**
  Imports all utility functions (e.g., constants like **DATA_TABLES**, **SCHEMA_WORKS**, **WAREHOUSE_TABLES**, and possibly **PIPELINE_RUNS_TABLE**).

Main Workflow (handler function)

1. **Initialization**
   - Prepares a results dictionary.
   - Iterates over each table in **DATA_TABLES**.
2. **Extraction**
   - Retrieves metadata about the last successful run for the table.
   - Calls **fetch_incremental_data** to get new or updated records since the last run.
   - Measures extraction duration and logs the number of records fetched.
3. **Metadata Preparation**
   - Prepares a metadata dictionary to track the status and metrics for each step.

- Updates metadata with data size, last record timestamp, and other relevant info.

4. **Transformation**
   - Selects the appropriate transformation function from `CUSTOM_TRANSFORMATIONS`.
   - Applies the transformation to the extracted data.
   - Each transformer function is responsible for cleaning, normalizing, or restructuring the raw data according to the needs of the target BigQuery schema.
   - The transformer must return a tuple: (ok, load_ready_data), where ok is a boolean indicating success, and load_ready_data is the transformed data (typically a list of dictionaries).
   - If transformation fails (ok is False), the pipeline logs the failure and skips to the next table.
   - Updates metadata with transformation status and duration.
5. **Schema Resolution**
   - Ensures the transformed data matches the expected schema (`SCHEMA_WORKS`).
   - Adds missing columns with `None` values.
   - Removes unexpected columns to prevent load failures.
6. **Loading**
   - Initializes a BigQuery client.
   - Loads the cleaned and transformed data into the appropriate BigQuery table (`WAREHOUSE_TABLES`).
   - Updates metadata with load status and duration.
7. **Logging and Notification**
   - Logs the pipeline run metadata using `log_pipeline_run`.
   - Sends a summary message using `google_worksspace_message`.
8. **Finalization**
   - After processing all tables, prints a completion message and returns the results.

Error Handling

- If any step fails (extraction, transformation, loading), the error is logged, and the pipeline continues with the next table.
- Metadata is updated with failure reasons for traceability.

Summary

This service automates the process of extracting, transforming, and loading data for multiple tables, ensuring schema consistency and robust error handling. It is modular, with table-specific transformations and clear separation of concerns for each ETL stage. The use of metadata and notifications provides transparency and monitoring for each pipeline run.