JSON to Snowflake Data Pipeline - Documentation

1. Overview

This document outlines the architecture, implementation, and design decisions for the JSON to Snowflake data pipeline. The pipeline processes restaurant data from a JSON source, performs validation and transformation, and loads it into a Snowflake data warehouse. The process is scheduled to run weekly using Apache Airflow.

2. Data Sources

• Source: A JSON file containing city-wise restaurant data from Swiggy.
• Nature: Static data updated once per week.
• Format: Nested JSON where each city contains a dictionary of restaurants with associated metadata (ID, name, rating, cuisine, cost, etc.).

3. Data Transformation

• Normalize the data into a flat structure suitable for Parquet format.
• Consistent date formats and field types.
• Handle missing or invalid values (e.g., 'NA', '--') with default strategies.
• Deduplicate rows by restaurant ID.
• Validation with Pandera schema enforcement.
• Generate summary CSV logs of transformed files and validation errors.

4. Data Loading

• Upload Parquet chunks to Amazon S3.
• Use Snowflake external stage to load data using COPY INTO.
• Data is inserted into a structured `restaurants` table in Snowflake.
• Metrics are logged (rows loaded, time taken).
• Backup strategy: Previous table data is backed up before load into a dedicated backup table.

5. Data Integrity

• The order of chunk upload and COPY INTO ensures no race condition.
• Dual logging to local `logs/` and Airflow UI for observability.
• Invalid rows are logged with complete field trace for review.
• Source JSON is archived as a ZIP after successful ingestion.

6. Performance Optimization

• Use of Parquet format for efficient I/O.
• Chunk-based processing to minimize memory usage.
• Use of PyArrow and Pandas for fast transformation.

• Load performance metrics are recorded into CSV logs.
• File deletion/archival logic to maintain storage hygiene.

7. Automation and Scheduling

• Entire pipeline orchestrated with Apache Airflow.
• Scheduled to run weekly using cron `0 3 * * SUN`.
• DAG steps:
  1. Ingest JSON → Parquet
  2. Transform Parquet
  3. Upload to S3
  4. Load to Snowflake
  5. Archive uploaded Parquet files in S3

8. Assumptions Made

• Restaurant data is uniquely identified by the `id` field.
• JSON schema remains stable week to week.
• External stage `swiggy_stage` is pre-configured in Snowflake.
• AWS and Snowflake credentials are managed via .env file.
• Weekly full ingestion is sufficient to capture all changes.
• Errors will be handled through logging rather than retries.

9. Project Structure

```
src/
├── ingestion/ingest_json_to_parquet.py
├── transformation/transform_parquet_data_json.py
├── s3_upload/upload_to_s3_json.py
├── loading/load_to_warehouse_json.py
├── archiving/archive_json_files_in_s3.py
├── utils/logger.py

dags/
└── pipeline_json_dag.py

tasks/
├── ingest_json_task.py
├── transform_json_task.py
├── upload_json_task.py
├── load_json_task.py
└── archive_json_task.py
```