

Sure! Here's a step-by-step PDF explanation of your Python and Bash pipeline, leaving out implementation notes, pitfalls, and suggestions.

PSV → Parquet Processor: Step-by-Step Guide

1. Overview

The pipeline processes PSV (pipe-separated values) files containing observational data and converts them into Parquet format. The workflow supports batch processing, chunked reading, and parallel jobs with logging.

2. Python Code Breakdown

a. Schema Definition

- `SCHEMA_DTYPES` defines the expected columns and data types.
- `COLUMN_ORDER` ensures consistent column ordering for Parquet output.

b. Helper Functions

- `enforce_types(df)`: Converts each column to its designated type.
- `format_seconds(seconds)`: Formats elapsed time into HH:MM:SS.
- `get_file_tag(timestamp, freq)`: Creates a file tag (YYYY_MM) based on the timestamp.
- `append_to_parquet(table, pq_path)`: Writes a PyArrow table to Parquet. If the file exists, it reads, combines, and rewrites the table.

c. Batch Processing

- `process_batch(psv_files, output_dir, freq, log_dir, chunksize)`: Reads PSV files in chunks, enforces types, groups by file tag, and appends to Parquet.

d. Dynamic/Adaptive Batching

- `create_dynamic_batches(station_files, input_dir, memory_safety)`: Groups station files into memory-safe batches based on file size and available RAM.

e. Main Processing Function

- `process_all_stations(station_files, input_dir, output_dir, freq, batch_size, log_dir, job_id, chunksize)`:
 - Creates a job-specific temporary directory.
 - Generates dynamic batches.
 - Processes each batch via `process_batch`.
 - Logs total rows processed and elapsed time.

f. Merge Function

- `merge_chunks(temp_dir, final_dir, freq, log_dir)`:
- Collects all temporary Parquet files.
- Groups by file tag.
- Concatenates tables and writes final Parquet files.
- Logs the total merged rows.

g. Command-Line Interface

- Accepts arguments for frequency (`--freq`), input/output directories, batch size, job ID, number of jobs, chunk size, and merge-only mode.
- Handles job splitting by `job_id`.
- Supports automatic merge after all jobs finish.

3. Bash Submit Script Breakdown

a. Configuration

- `NUM_JOBS`: Number of parallel jobs.
- `CHUNK_SIZE`: Rows per chunk.
- `FREQ`: Frequency of data aggregation (`sub_daily`, `hourly`, etc.).
- Directories: `INPUT_DIR`, `OUTPUT_DIR`, `FINAL_DIR`, `LOG_DIR`.
- `STATION_LIST`: File listing all station PSV files.
- `PYTHON_SCRIPT`: Path to the Python processor.
- `PYTHON_ENV`: Path to Python environment.

b. Directory Setup

- Ensures that `LOG_DIR`, `OUTPUT_DIR`, and `FINAL_DIR` exist using `mkdir -p`.

c. Launching Parallel Jobs

- Loops over job indices from 0 to `NUM_JOBS-1`.
- Each job calls the Python script with its unique `--job_id` and `--num_jobs` for partitioning.
- Background jobs are run using `&`.
- `wait` ensures the script waits for all jobs to finish.

d. Merging Final Parquet Files

- After all jobs complete, the Python script is called in `--merge_only` mode.
- Consolidates temporary Parquet outputs into the final directory.

e. Output

- Prints progress messages and completion notices.
- Logs total rows processed and merge completion.

This PDF explains how the PSV → Parquet pipeline operates in a structured, step-by-step manner without implementation notes, pitfalls, or suggestions.