PSV to Parquet Processing Workflow

1. Python Script Summary

Purpose:
- Convert .psv.gz files (daily/sub-daily observations) to Parquet.
- Supports chunked reading, append-safe writes, dynamic/adaptive batching, parallel jobs, and merging.

Key Features:
- Chunked reading: Reads in CHUNK_SIZE rows to limit memory usage.
- Append-safe Parquet: Safely combines chunks into same Parquet file.
- Dynamic/adaptive batching: Groups files based on size and available RAM.
- Job splitting: Stations distributed across NUM_JOBS via job_id.
- Logging: Tracks rows processed and failures.
- Merge: Combines temporary Parquet files into final output.

2. Bash Submit Script Summary

Purpose:
- Launch multiple parallel Python jobs safely.
- Trigger final Parquet merge after all jobs.

Key Features:
- Parallel jobs: Loops over NUM_JOBS and runs Python jobs in background.
- Safe RAM usage: Relies on Python script's dynamic batching.
- Merge: Triggered after all jobs to combine temporary Parquet files.
- Directories: Ensures log, temp, and final output directories exist.

3. Overall Workflow

1. Bash script prepares environment and launches NUM_JOBS parallel Python jobs.
2. Each Python job:
   - Reads assigned stations/files.
   - Dynamically batches files based on size/memory.
   - Reads .psv.gz in chunks.
   - Cleans, types, groups by file_tag, and writes/appends to Parquet.
   - Logs rows processed and failures.
3. Parallelism: Multiple jobs run simultaneously.
4. Post-processing: Python script merges temporary Parquet files into final directory.
5. Cleanup: Temporary job folders deleted.
6. Result: Final, aggregated Parquet files by freq and YYYY_MM.

4. Advantages

- Memory-safe: Handles large files without swapping.
- Automatic batching: No manual tuning.
- Parallel execution: Efficient CPU usage.
- Resilient: Logs failed rows, merges results automatically.
- Scalable: Works for hundreds of stations and hundreds of GBs of data.