How would you optimize, benchmark, and scale this pipeline to thousands of samples?

Optimizations:

- Avoid loading full gVCF files into memory, as raw gVCFs can contain records for every genomic position. Instead, implement a streaming parser to clean data prior to dataframe construction.
- Explore using a high-performance dataframe library (e.g. Polars), which has improved performance over Pandas, to increase filtering and aggregation speed.
- Revisit file I/O (input/output) strategy, as compressed parsing and disk access can become bottleneck as input scales.
- Include input validation or integrity checks to prevent silent downstream errors.

Benchmarking:

- Benchmark on representative dataset sizes (10, 100, 1000+ samples).
- Measure runtime and resource usage (CPU, memory) per sample or per stage (parsing, filtering, aggregation, output) to identify bottlenecks and inform batching strategy.

Scaling:

- Parallelize at the sample level using HPC job-arrays or task farming, processing samples independently or in batches.
- Implement structured logging (rather than terminal-only output), to enable monitoring and automated detection/resubmission of failed samples.
- Use a robust dependency/environment manager (e.g. conda or uv), instead of requirement.txt, to ensure reproducibility as pipeline complexity grows.
- Integrate into a workflow manager (e.g. Nextflow) with containerized execution (e.g. Docker) to enable reproducible scaling across HPC or cloud environments.