Batch Consumer Requirements – Interview Cheat Sheet

What you own as the batch consumer / pipeline team

• Define batch cadence and SLAs (hourly/daily; cutoff times; downstream availability).

• Define the watermark strategy (how you detect new/changed data) and lateness tolerance.

• Ensure idempotent, rerunnable loads (reprocess partitions safely without duplicates).

• Design incremental processing (process only new/changed partitions; avoid full refresh).

• Enforce schema + evolution (versioned schemas; compatibility checks; controlled rollouts).

• Data quality gates (counts, nulls, ranges, referential integrity; warn vs block).

• Backfill and restatement strategy (parameterized runs; historical recompute).

• Observability + SLAs (job duration, success rate, completeness, freshness, cost).

Minimum metadata to capture for batch runs

Metadata	Why it matters
run_id / batch_id	Traceability and debugging; ties outputs to a run
input_snapshot_time / watermark	Defines which data was included; reproducibility
source_system / dataset	Lineage and ownership
partition keys (e.g., business_date)	Incremental loads and safe reruns
record counts (in/out)	Completeness monitoring and reconciliation
checksum / file list (optional)	Proves exact inputs used; audit and replay
schema_version	Controlled schema evolution

Key SLIs to monitor (tie to SLAs)

• Availability time: when the curated dataset is published (e.g., by 7am)

• Job duration: runtime vs SLA; queue time and retries

• Success rate: % successful runs over a window

• Freshness: now – max(business_time) loaded (or watermark delay)

• Completeness: expected vs actual partitions/rows; late-arrival volume

• Quality: null rates, range violations, referential integrity failures

• Cost: compute hours, $/run, $/TB processed

Reliability patterns (batch-specific)

• Idempotent writes via overwrite-by-partition or MERGE with deterministic keys

• Atomic publish pattern: write to staging then swap/commit to production table

• Rerunnable partitions: rerun a date range safely with no duplicates

• Backfills/restatements: parameterized jobs; isolate historical recompute

• Late data handling: reopen partitions within a defined window; track late-arrival metrics

• Failure isolation: retry transient steps; quarantine bad input files/records

Ownership split (platform/producer vs batch pipeline team)

Owned by platform/producer (mostly)	Owned by batch pipeline team (you)
Providing source extracts/snapshots; source SLAs; upstream schema changes	Incremental logic, idempotent loads, partition strategy, schema en
Delivery of files/exports and retention	Run orchestration, backfills, reconciliation, quality checks, publish
Access controls at source	Observability, cost controls, downstream SLAs and consumer com

30-second spoken answer (memorize)

“For batch pipelines, I first confirm the cadence and SLAs—when the dataset must be available—and define a watermark strategy to detect new or changed data. I design incremental processing so we only load impacted partitions, and I make jobs idempotent and rerunnable using overwrite-by-partition or MERGE with deterministic keys. I enforce schema evolution through versioned schemas and compatibility checks, and I apply data quality gates with clear actions—warn, quarantine, or block. I also define a backfill/restatement approach for historical corrections. Finally, I monitor SLIs like availability time, duration, success rate, freshness, completeness, and cost against SLAs.”