

WORKFLOW REVIEW / DECISION BRIEF

Orders pipeline reliability review

One workflow in BigQuery and Composer/Airflow

This example shows the findings and decisions a focused review can leave with your team. The company, workflow and records are fictional. Local test results come from the accompanying Python rehearsal; they are not client results, a cloud benchmark or a promised saving.

RECOMMENDED DECISION

Correct the retry behaviour and extraction rule before trusting this reporting source. The local candidate handles the supplied source versions correctly. Production should wait until the source owner confirms the update contract and the candidate passes tests in the actual BigQuery and Composer environment.

The reporting problem

A daily orders table feeds a morning operations report. An incremental task selects orders by event date and appends the result. Repeating a batch creates a duplicate; changes to older orders are missed. A successful task status alone would not expose either error.

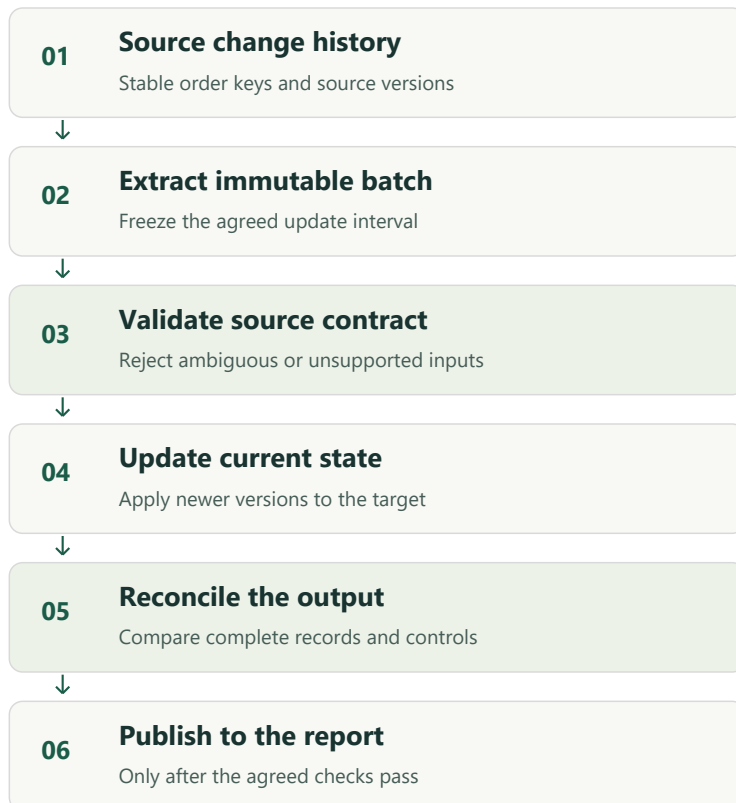
LOCAL FIXTURE RESULT	UNSAFE BASELINE	EXPECTED AND CANDIDATE
Rows in current output	5	4
Distinct order keys	4	4
Amount control total	GBP 1,300.00	GBP 1,020.00
Late corrections to A and B	Missing	Present

The amounts are fictional reconciliation controls. Their difference is not measured lost revenue, cost reduction or commercial impact.

02 / REVIEW BOUNDARY

Scope and workflow

The review follows one orders workflow from source changes to its current-state output. The diagram shows the intended data path; the rehearsal models its behaviour locally.



Illustrative boundary

One daily workflow, two change batches, one target table and four order keys. Six source versions are supplied across the initial snapshot and batches.

Source assumptions

Stable keys, UTC update timestamps and ordered source versions. Each version has one complete payload. **Deletes are not supported** by this candidate.

Independent reference

The expected state is specified independently in **expected.json**. It is not computed by the candidate update function.

Composer/Airflow would orchestrate these steps in a real environment. This example does not create an Airflow deployment or execute SQL on BigQuery.

Evidence needed from a real environment

Start with redacted code, table definitions, agreed source rules and selected run or job metadata. Link the affected task runs to their BigQuery jobs. The actual billing model, region, freshness requirement and business control totals determine whether a change is worthwhile.

Client-run exports are an option. If direct access is necessary, agree its purpose, scope, expiry and removal first. No credentials or confidential records are needed for an initial enquiry. A review does not authorise production writes.

Outside this review: the entire warehouse, every DAG, the reporting tool and the security of a Google Cloud organisation.

03 / WHAT TO FIX FIRST

Findings and recommended actions

F01

Repeating a batch duplicates an order

PRIORITY
HIGH

E03 contains two copies of order C after the same event-day batch is appended twice. T01 detects the invalid baseline. A task retry must not create a new business record merely because the previous acknowledgement was lost.

Recommended action: Use a validated current-state update keyed by order_id and source_version. T03, T07 and T15 check replay, identical source duplicates and simulated lost-acknowledgement recovery. Actual SQL, orchestration and concurrency still need testing.

EVIDENCE E03 / CHECKS T01, T03, T07, T15

F02

An event date filter misses later corrections

PRIORITY
HIGH

In E01, A and B change after their original event days. An event-day-only extraction misses those versions. E02 identifies the corrected expected amounts and T05 confirms both are present in the candidate.

Recommended action: Use a cursor supported by the source contract: reliable update timestamps or a change-data-capture position. Use explicit intervals and immutable batches. Agree lateness, overlap and reconciliation; a lookback interval chosen without evidence cannot guarantee completeness.

EVIDENCE E01, E02 / CHECK T05

F03

A task success does not establish data correctness

PRIORITY
HIGH

A pipeline can finish with duplicate keys or stale amounts. The example uses exact row comparison as well as key and amount checks, so an unchanged aggregate alone cannot mask a wrong record.

Recommended action: Define the reference period, ownership and acceptance tests before a change. Keep the previous published output available until the candidate passes. Mismatches and unknown source rules are release blockers.

EVIDENCE E02, E04, E05 / CHECKS T02, T18

04 / PROOF AND LIMITS

Validation results

20 / 20

LOCAL CHECKS PASSED

Python correctness checks
Cloud validation remains pending

The expected output is a literal fixture authored independently of the update function. These results demonstrate the contracted behaviour for the supplied cases; they do not prove correctness for every source or production workload.

CHECKS	RESULT	BEHAVIOUR COVERED
T01 to T02	PASS	Reject the unsafe baseline and match the independent expected snapshot.
T03 to T07	PASS	Replay, older versions, late corrections, reversed arrival and identical duplicates.
T08 to T12	PASS	Block conflicting versions, null keys, invalid timestamps, deletes and schema drift.
T13 to T16	PASS	Empty batches, failure before publication, lost acknowledgement and local rollback.
T17 to T20	PASS	Interval boundaries, exact reconciliation, whole-batch validation and timestamp ordering.

Production acceptance plan

- 01 Confirm the source contract and capture a representative independent reference snapshot, including late data and the agreed deletion policy.
- 02 Execute adapted SQL in an isolated client dataset. Compare keys and values, replay batches and test invalid inputs. Set a query cost envelope first.
- 03 Exercise the installed Composer/Airflow version, fixed intervals, immutable batches and lost-acknowledgement recovery. Enforce a single writer or test an agreed concurrency design.
- 04 Rehearse restoration of data, checkpoint and known-good code together. Record the approver, recovery owner, downstream publication and observation window.

Not yet tested BigQuery SQL, Composer/Airflow scheduling, live extraction, concurrent writers, production restoration, runtime and cloud charges. SQL files are design templates. A real implementation needs its own agreed scope and acceptance record.

05 / REVIEW TRAIL

Evidence and ownership

The technical appendix makes the local reasoning inspectable. Each item is traceable below; the roles are proposed responsibilities for a real engagement.

ID	FILE OR RECORD	WHAT IT ESTABLISHES
E01	fixtures.json	Fictional initial snapshot and two immutable change batches.
E02	expected.json	Independent state: A 110.00, B 210.00, C 300.00 and D 400.00 GBP.
E03	evidence/baseline.json	The reproduced duplicate and stale values in the unsafe approach.
E04	evidence/candidate.json	Candidate output compared with the independent reference.
E05	evidence/checks.json	Result and meaning of each of the 20 executed local checks.
E06	SQL templates and RUNBOOK.txt	Proposed cloud validation and recovery plan; execution remains pending.

Proposed ownership in a real engagement

OWNER	DECISION OR RESPONSIBILITY
Source owner	Confirm key, version, update, deletion and late-arrival semantics.
Data workflow owner	Approve the acceptance reference, test environment and operational handover.
Deployment approver	Authorise the specific production change, release window and rollback.
AgileDataWarehouse	Deliver the agreed review or change, evidence, limitations and handover notes.

Run the local example

Extract the appendix and run **python run_rehearsal.py** with Python 3.10 or later. It uses only the standard library. No cloud account or credentials are required.

06 / THE NEXT DECISION

From findings to a scoped next step

Your team can act on the report without committing to a larger project. A review is useful when it resolves uncertainty that matters to the next decision.

01 Implement with your own team

Use the evidence, proposed correction and acceptance plan as the basis for an internally owned change.

02 Request a focused implementation

Agree the selected correction, tests, access, production authorisation, rollback and handover in a separate proposal.

03 Clarify the source contract first

Stop before building if update or deletion semantics remain uncertain. Resolve those rules with the source owner.

Planned ongoing care is optional. It may cover agreed workflows, checks and improvements within reserved capacity. It is not automatically included in a review or a substitute for 24/7 incident cover.

Interpretation and method

No saving, runtime improvement, service level or incident frequency is estimated from six synthetic source versions. An unfiltered target MERGE may scan substantial data. Representative query plans, workload measurements and actual billing are needed before recommending it for efficiency.

Airflow recommends repeatable task outcomes across retries. BigQuery documents atomic MERGE behaviour; this does not make extraction and orchestration atomic. Actual environment and versions govern the final design.

[Airflow task design guidance](#) / [BigQuery MERGE reference](#)

Discuss one workflow

Describe what runs, what goes wrong and what a useful first result would be. Start by email or the website form. Scope, access, price and timing are agreed in writing before work begins. Relevant professional details are shared privately before contracting.

[Describe your workflow at agiledwh.com](#)