

## Title

**\*\*The Delivery Challenge\*\***

## Persona

**\*\*You are Lemlem\*\***, a data team member building a client's data warehouse.

## Executive Summary

Lemlem discovered **\*\*non-trivial reconciliation gaps\*\*** between Finance and Delivery during proactive validation. Because the program is on a fast-track schedule with no validation buffer, the immediate goal is to secure **\*\*client agreement for a time-boxed investigation\*\*** to isolate whether gaps are caused by data quality, timing/reporting windows, or definition mismatches-before these issues become embedded in the warehouse as "truth."

This report includes a **\*\*client-ready update\*\*** that balances urgency with diplomacy, proposes a concrete and limited scope, and requests explicit approval to allocate time to root-cause the discrepancies.

## Scenario

Lemlem is a recently appointed mid-level Data Engineer tasked with spearheading the development of a centralized data warehouse for a primary client. As a new addition to the team, Lemlem faces the "Liability of Newness"-a sociological phenomenon where a lack of established trust and institutional knowledge complicates the navigation of internal politics. This role serves as Lemlem's debut in high-stakes client delivery, requiring a delicate balance between technical rigor and executive diplomacy.

The project environment is characterized by a significant internal schism regarding digital transformation. While senior leadership views the centralized warehouse as a catalyst for organizational scaling, the operational staff exhibits varying degrees of "information hoarding." This defensive posture suggests a perceived threat to departmental autonomy or a fear of exposure regarding legacy process inefficiencies. Consequently, data discovery is hindered by stakeholders who are protective of their datasets and resistant to cross-functional inquiry.

The project is currently operating under a compressed "fast-track" schedule with zero allocated buffer for iterative validation. The discovery of discrepancies between Finance and Delivery datasets introduces a critical project management dilemma. Because the project plan is rigid, the effort required for a comprehensive root-cause analysis represents a zero-sum trade-off. Lemlem must navigate the "Project Management Triangle," where the pursuit of data integrity (Quality) directly threatens either the project timeline (Time) or the team's human capital (Scope/Burnout), necessitating a strategic decision on resource prioritization.

During validation that you undertook on your own, you discover some curious discrepancies between finance data and delivery data. Totals and counts do not fully reconcile.

The root cause is unclear. Possible explanations include:

- \* Data quality issues
- \* Timing/reporting differences
- \* Definition mismatches
- \* Team misunderstanding

- \* Something nefarious
- \* System inconsistencies

There is no evidence of wrongdoing, yet.

Your manager asks you to draft a client update to ask for agreement to allocate some time to look into the discrepancies.

Task 1: Data Analysis (Findings from the sample extracts)

I reviewed the sample Finance payments extract against the Delivery status extract to understand how the two datasets relate and what could prevent reconciliation at scale.

**Note:** This section is **internal working analysis** (not intended to be sent to the client). The client-facing content is in Task 2.

### 1) Duplicate or replayed Finance records (double-count risk)

Some orders/packages appear twice in Finance with the same amount/date/status, which inflates totals if summed naively.

\* Example patterns observed: same 'order\_id' + 'package\_id' + 'payment\_date' + 'amount' + 'payment\_status' repeating with different 'payment\_id'.

### 2) Missing or incomplete join keys (unmatchable records)

Finance contains at least one row where 'package\_id' is blank. If Delivery joins rely on 'order\_id + package\_id', these rows cannot reconcile until the key is corrected or mapped.

### 3) Timing / reporting window differences (month boundary drift)

Several deliveries occur days later (e.g., paid in late January, delivered in early February). Monthly reporting will diverge if Finance is grouped by 'payment\_date' while Delivery is grouped by 'event\_time'.

### 4) Definition mismatches (paid != delivered)

Delivery includes non-delivery outcomes ('FAILED', 'RETURNED'). If Finance reports "PAID" and Delivery reports "DELIVERED," these won't tie out without agreed business rules (e.g., whether to exclude returns, whether failures are re-shipped, etc.).

### 5) Lifecycle gaps (present in one system, missing in the other)

The sample includes payments for orders/packages that do not appear in the Delivery extract and vice versa (depending on extract completeness). In a warehouse, we need to know whether that represents:

- \* late-arriving data,
- \* true process exceptions,
- \* or extract/ETL coverage gaps.

**Conclusion:** These are all plausible "normal" causes in legacy systems, but they must be explicitly resolved and documented so the warehouse metrics are trusted and repeatable.

Task 2: Communication (client-ready message contents, <=400 words)

**Subject:** Request to time-box Finance vs Delivery reconciliation before warehouse sign-off

Hi [Client Team],

As part of validation for the centralized data warehouse, we ran an initial reconciliation

between the Finance payments extract and the Delivery status extract. We're seeing **\*\*differences in totals and counts\*\*** that prevent a clean tie-out between "paid" activity in Finance and "delivered" outcomes in Delivery.

At this stage, the root cause is **\*\*not yet confirmed\*\***. Based on what we see so far, the gaps are consistent with common legacy-system patterns such as:

\* **\*\*Duplicate/replayed payments\*\*** (same order/package/amount/date appearing more than once), which can inflate Finance totals if not deduplicated.

\* **\*\*Missing or inconsistent join keys\*\*** (e.g., a blank 'package\_id'), which makes some Finance records unmatchable to Delivery.

\* **\*\*Timing differences\*\*** (e.g., paid late in month, delivered early next month), which causes month-end drift depending on which timestamp is used.

\* **\*\*Definition differences\*\*** (e.g., Finance "PAID" vs Delivery outcomes like 'FAILED'/'RETURNED'), which require agreed business rules for reporting.

To avoid locking incorrect assumptions into the warehouse, we recommend a **\*\*time-boxed reconciliation investigation (1-2 business days)\*\*** to produce a short decision memo with:

- 1) agreed metric definitions ("paid," "delivered," "returned/refunded," etc.),
- 2) deduplication + join keys/rules,
- 3) a small set of known exceptions (and how they will show up in dashboards).

**\*\*Options:\*\***

\* **\*\*Option A (recommended):\*\*** Approve the 1-2 day time-box now to confirm root causes and finalize rules before we publish executive metrics.

\* **\*\*Option B:\*\*** Continue on the current schedule with provisional rules (higher risk of later rework and reporting disputes).

**\*\*Recommendation:\*\*** Option A. This is the smallest investment that materially reduces the risk of stakeholder misalignment later.

Please confirm whether we can proceed with **\*\*Option A\*\*** and identify 1 Finance + 1 Delivery point person for a short working session. If approved, we will share the memo and implementation rules immediately after the time-box ends.

Thanks,  
Lemlem  
Data Engineering

### Simple diagram (how reconciliation breaks)

```
""
Finance (payments) --(order_id + package_id + rules)--> Delivery (status events)
    | duplicate rows / missing keys / timing drift / status definitions |
    +----- causes mismatch -----+
""
```

Materials Provided

You receive:

\* Client brief (above)

\* Sample data (extracts)

\* Guidance on desired next actions

\* (Assume materials may be imperfect or incomplete - like real projects.)