Custody Data Governance – End-to-End Project Documentation

# Author & Date

Author: Priyanka P   
Date: September 16, 2025

# Overview

This document provides a complete, step-by-step explanation of the Custody Data Governance project. It covers scope, architecture, data domains, data quality (DQ), lineage (business, technical, and column level), security (including Row-Level Security for Power BI), governance catalog (CDEs & ADS), monitoring.

# Objectives

1. Establish a governed, auditable data flow for Custody domain (Accounts, Transactions, Securities, Corporate Actions).  
2. Enforce data quality and promote only verified records from Staging to Production.  
3. Provide transparent lineage at business, technical, and column levels for compliance and stakeholder trust.  
4. Enable secure, role-based analytics in Power BI with live SQL connection and Row-Level Security (RLS).

# Architecture

Landing/Raw → Power Query (ETL) → SQL Server Staging → DQ Enforcement → SQL Server Production → Power BI (Live) → Dashboards/RLS  
- Landing: CSV/Excel drop or file share.  
- Power Query: Cleanse, standardize, type-cast, basic validations, tag issues.  
- Staging schema: Holds cleansed-but-unverified data.  
- DQ Enforcement: SQL constraints, checks, and stored procedures to validate business rules.  
- Production schema: Stores verified and curated data for BI.  
- Power BI: Live connection, semantic model, RLS, KPIs and drilldowns.

# Data Domains & Critical Data Elements (CDEs)

Domains: Accounts, Transactions, Securities, Corporate Actions.

Representative CDEs:  
- Account\_Number  
- Security\_ID  
- Trade\_Date  
- Settlement\_Date  
- Trade\_Amount (derived as Quantity \* Price)  
- Quantity, Price, Currency\_Code, Status

Each CDE is registered with definition, owner/steward, datatype, authoritative data source (ADS), quality dimensions, and sensitivity classification.

# Schemas & Tables (SQL Server)

Schemas: staging, prod

Core tables:  
- staging.Accounts, staging.Transactions, staging.Securities, staging.CorporateActions  
- prod.Accounts, prod.Transactions, prod.Securities, prod.CorporateActions

Views (for BI-friendly modeling):  
- prod.vw\_Fact\_Transactions (adds Settlement\_Days; exposes Trade\_Amount)  
- prod.vw\_Fact\_CorpActions  
- prod.vw\_Dim\_Accounts, prod.vw\_Dim\_Securities

# Data Quality (DQ) Framework

Quality dimensions applied: Completeness, Consistency, Conformity, Uniqueness, Accuracy, Timeliness.

Examples:  
- Completeness: Account\_Number, Security\_ID, Trade\_Date must be NOT NULL.  
- Consistency: Settlement\_Date >= Trade\_Date.  
- Conformity: Currency\_Code in ISO list; Status in ('Filled','Cancelled','Pending').  
- Uniqueness: Natural/business keys not duplicated (e.g., Transaction\_ID unique).  
- Accuracy: Trade\_Amount = Quantity \* Price.  
- Timeliness: Reject records older/newer than policy thresholds.

Enforcement: CHECK constraints, UNIQUE indexes, stored proc validations, error quarantine tables, and metrics tables for monitoring.

# Lineage Levels & Types

Levels:  
1) Business lineage (systems & flows): Landing → Staging → Prod → Power BI  
2) Technical lineage (jobs, tables, views, ETL): staging.\* → DQ → prod.\* → views → PBI dataset  
3) Column/attribute lineage: e.g., prod.Transactions.Trade\_Amount = staging.Transactions.Quantity \* staging.Transactions.Price  
4) Code/logic lineage (advanced): SQL/Power Query/DAX and any scripts tracked in Git.

Types:  
- Manual: Excel/CSV mapping and diagrams.  
- Automated: Tool-scanned metadata (e.g., Apache Atlas, Collibra) where available.  
- Hybrid: Automate common flows; document exceptions manually.

# Security & RLS (Power BI)

- Principle of Least Privilege.  
- RLS example: Restrict rows by OwnerEmail = USERPRINCIPALNAME().  
- Sensitive fields: mask or exclude depending on role.  
- Audit access via BI workspace permissions and SQL roles.

# Governance Catalog (CDE & ADS)

Register:  
- CDEs (name, definition, datatype, owner/steward, sensitivity, quality dimensions).  
- ADS (authoritative data sources): system-of-record for specific CDEs.  
- Ownership model: Data Owners (accountable), Data Stewards (responsible), Custodians (technical), Consumers (analysts).

Maintain change logs and periodic reviews.

# Monitoring & Alerts

- DQ metrics tables compute pass/fail counts and severity by rule.  
- Threshold-based alerts (e.g., email or Teams via scheduled job) when failure rate exceeds limits.  
- Trend reporting in Power BI for DQ and lineage coverage.

# Testing Strategy

- Unit tests: SQL stored proc logic, CHECK constraints.  
- Integration tests: File → Staging → DQ → Prod flow with sample data.  
- BI validation: Aggregations, totals, and drilldowns reconcile to source.  
- Security tests: RLS scenarios by role.

# Change Management

- Git-based version control for SQL, Power Query M scripts, and documentation.  
- Tag releases for each BI dataset refresh change.  
- Data contract with source teams: column changes require notice and test plan.

# Risks & Mitigations

- Incomplete lineage: Mitigate with mandatory mapping templates and code reviews.  
- Schema drift: Detect via automated schema checks and break-glass alerts.  
- Performance bottlenecks: Index tuning, partitioning strategy for large tables.  
- Access sprawl: Quarterly access recertification and RLS audits.

# Glossary

- Data Governance: People, processes, and technologies to ensure data is accurate, secure, and usable.  
- Data Lineage: The life story of data—where it came from, how it moved, and how it changed.  
- CDE (Critical Data Element): A data element critical to business processes or regulatory reporting.  
- ADS (Authoritative Data Source): The system recognized as the official source of a data element.  
- DQ Rule: A business/technical rule ensuring data meets quality thresholds.  
- RLS: Row-Level Security; restricts data rows based on the viewer’s role or identity.  
- Staging vs Production: Staging is pre-validated; Production is curated and certified for analytics.

# How-To: Step-by-Step Build

1. Prepare landing files (CSV/Excel) for Accounts, Transactions, Securities, Corporate Actions.  
   2) Use Power Query to standardize column names, types, trim/clean text, flag nulls, and export to SQL staging.  
   3) Create SQL schemas and tables for staging and prod.  
   4) Implement DQ: NOT NULL, CHECK constraints, and a stored procedure to validate complex rules; write failures to quarantine tables.  
   5) Promote passing records from staging to prod.  
   6) Create BI-friendly views in prod schema and publish Power BI semantic model with RLS.  
   7) Register CDEs & ADS in governance CSVs; assign owners/stewards and sensitivity.  
   8) Build lineage CSV (business, technical, and column mappings).  
   9) Schedule refresh and DQ monitoring jobs; wire alerts to email/Teams.  
   10) Version everything in Git (SQL, M scripts, docs, CSVs).

# Final flow

1. **Landing/Raw files (CSV/Excel) → DQ Pre-Checks**
2. **Power Query (ETL: cleanse, standardize, *tag issues*)**
3. **SQL DB → staging schema** (land cleansed-but-unverified data)
4. **Post-ETL DQ Enforcement → promote to prod schema**
5. **Power BI (live SQL) + RLS**
6. **Lineage Documentation (Visio/Lucidchart)**
7. **Governance catalog updates** (CDE, ADS, owners) + **Monitoring/alerts**

**What to do in each step (super short)**

**1) Landing + Pre-checks**

* Validate file structure, required columns, basic types.
* Quarantine bad files; log counts.

**2) Power Query**

* Standardize dates/currencies; trim/case; dedupe; derive fields.
* Add “DQ\_Flag” columns (e.g., IsFutureTrade, IsAcctLenBad) so you can show before/after.

**3) SQL staging**

* Load exactly-as-transformed rows; keep raw vs cleaned audit columns (\_ingested\_at, \_source\_file, \_dq\_flags).

**4) Post-ETL DQ in SQL (gate to prod)**

* Enforce business rules (10-digit accounts, non-future trade dates, FK integrity, non-negative amounts, Settlement ≥ Trade when Settled).
* Produce a **DQ scorecard** table + exceptions table.
* Only INSERT “pass” records to prod; keep “fail” in an exceptions table for RCA.

**5) Power BI (+ RLS)**

* Model off prod.\*.
* **RLS**: e.g., a role filter on Accounts[Region] to the user’s allowed regions.

**6) Lineage (attribute-level)**  
Show one column end-to-end (e.g., **Account\_Number**):  
CSV → PQ step list → staging.Accounts.Account\_Number → constraints/validations → prod.Accounts.Account\_Number → PBIX model → visual.

**7) Governance & Monitoring (nice bonus)**

* Register **CDEs** (Account\_Number, Security\_ID, Trade\_Date, Settlement\_Date, Trade\_Amount).
* Declare **ADS** per CDE.
* Nightly DQ job → email/Teams alert if thresholds breached; attach exceptions CSV.

**Security touches to mention**

* **Least-privilege** SQL roles; BI access via AD groups.
* **Mask PII** (e.g., show Investor\_Name masked in prod or at the BI layer).
* **Audit tables**: who loaded what, when, and counts of pass/fail.

**You’re not missing much, but these two extras impress:**

* **Exceptions/RCA loop** (ticket created per DQ breach; owner; fix; close).
* **DQ score** KPI on the dashboard (e.g., % rows passing critical rules).

# Step 1: Landing + Pre-checks (in Power Query)

using the dataset I generated. Below is a **click-by-click** plan you can follow in either **Power BI Desktop** or **Excel → Data → Get Data → Power Query**.

**0) Set up your Power Query project**

1. Open **Power BI Desktop** (or Excel).
2. **Get Data → Text/CSV** and import:
   * Accounts.csv
   * Securities.csv
   * Transactions.csv
   * Corporate\_Actions.csv
3. For each, click **Transform Data** (not Load).

Tip: In the **Queries pane**, create two query groups:

* **01\_Landing** (raw imports, no edits besides headers & types)
* **02\_PreChecks** (all validations, flags, and filtered “staging” + “exceptions” outputs)

**1) Create reusable Parameters (for rules)**

In **Power Query Editor → Home → Manage Parameters → New Parameter**:

* **pToday** (Date): = Date.From(DateTime.LocalNow())
* **pValidCurrencies** (Text List): = {"USD","EUR","GBP","CAD","MXN","AED","ZAR","INR","SGD","JPY","AUD","BRL","ARS","CLP","COP"}
* **pAccountLen** (Number): = 10

These keep your rules editable and interview-friendly.

# 2) Accounts – Landing & Pre-checks

Select **Accounts** query (move it into **01\_Landing**), then **Duplicate** it into **02\_PreChecks** as **Accounts\_Pre**.

**A) Clean + Types**

* **Use First Row as Headers** (if needed).
* Set types:
  + Account\_Number → Text
  + Investor\_Name → Text
  + Currency\_Code → Text
  + Account\_Type → Text
  + Country → Text
  + Region → Text
  + Open\_Date → Date
* **Transform → Format → Trim** and **Clean** on text columns.

**B) Rule flags (Add Column → Custom Column)**

Add these Boolean flags (M formulas):

* **IsAcctLen10**
* Text.Length([Account\_Number]) = pAccountLen
* **IsCurrency3**
* Text.Length([Currency\_Code]) = 3
* **IsCurrencyAllowed**
* List.Contains(pValidCurrencies, [Currency\_Code]?)

Create a **DQ\_Flag** (all must be true):

[IsAcctLen10] and [IsCurrency3] and [IsCurrencyAllowed]

**C) PK check & Duplicates**

* **Home → Keep Duplicates** on Account\_Number to create a temporary step; if any appear, branch this into **Accounts\_DuplicateIDs** (exceptions).
* Undo (Ctrl+Z) that step; then **Remove Duplicates** on Account\_Number in **Accounts\_Pre**.

**D) Split into Staging vs Exceptions**

* **Accounts\_Exceptions**: **Reference** Accounts\_Pre → filter DQ\_Flag = false. Add a **Reason** column (concatenate failed flags) using a Custom Column:
* Text.Combine(
* List.Select(
* {
* if not [IsAcctLen10] then "BadLength" else null,
* if not [IsCurrency3] then "BadISO3" else null,
* if not [IsCurrencyAllowed] then "NotAllowed" else null
* },
* each \_ <> null
* ),
* "|"
* )
* **Accounts\_Staging**: **Reference** Accounts\_Pre → filter DQ\_Flag = true. Keep business columns (drop the flag columns).

**3) Securities – Landing & Pre-checks**

Duplicate into **Securities\_Pre**.

**A) Clean + Types**

* Security\_ID (Text), Security\_Name (Text), Asset\_Type (Text), ISIN (Text), CUSIP (Text).
* Trim/Clean text.

**B) Minimal rules**

* **IsSecIdPresent**:
* not Text.IsNullOrEmpty([Security\_ID])
* (Optional) **IsCUSIPLen9** / **ISINLen12** if you’d like strict formats.

**C) Duplicates & splits**

* Create **Securities\_DuplicateIDs** (exceptions) by Keep Duplicates on Security\_ID.
* **Securities\_Staging**: remove duplicates, filter [IsSecIdPresent] = true.

**4) Transactions – Landing & Pre-checks**

Duplicate into **Transactions\_Pre**.

**A) Clean + Types**

* Set types:
  + Transaction\_ID (Text)
  + Account\_Number (Text)
  + Security\_ID (Text)
  + Trade\_Date (Date)
  + Settlement\_Date (Date, allow nulls)
  + Quantity (Whole Number)
  + Price (Decimal)
  + Trade\_Amount (Decimal)
  + Status (Text)
  + Currency\_Code (Text)
* **Detect type errors**: in Applied Steps, right-click the **Changed Type** step → **Keep Errors** to create a **Transactions\_TypeErrors** exceptions query. Then revert to the good branch and continue.

**B) Rule flags (Add Column → Custom Column)**

* **IsFutureTrade**
* [Trade\_Date] > pToday
* **IsSettledBad**
* ([Status] = "Settled") and ( [Settlement\_Date] = null or [Settlement\_Date] < [Trade\_Date] )
* **IsAmtNonNeg**
* [Trade\_Amount] >= 0
* **IsQtyPositive**
* [Quantity] > 0
* **IsAcctLen10**
* Text.Length([Account\_Number]) = pAccountLen
* **IsCurrency3**
* Text.Length([Currency\_Code]) = 3

**C) Duplicate Transaction\_ID detection**

1. **Reference** Transactions\_Pre as Transactions\_IDCounts.
2. **Group By** Transaction\_ID → **Count Rows** as TxnCount.
3. Merge Transactions\_IDCounts back into Transactions\_Pre on Transaction\_ID, expand TxnCount, and create:
   * **IsTxnUnique**:
   * [TxnCount] = 1

**D) FK checks (merge with staging parents)**

* **Merge** Transactions\_Pre with **Accounts\_Staging** on Account\_Number (Left Outer), expand a dummy field (e.g., Investor\_Name) to test presence, add:
  + **HasAccount**:
  + [Investor\_Name] <> null
* **Merge** with **Securities\_Staging** on Security\_ID, expand Security\_Name, add:
  + **HasSecurity**:
  + [Security\_Name] <> null

**E) Build DQ\_Flag & split outputs**

* **DQ\_Flag\_Transactions**:
* (not [IsFutureTrade]) and (not [IsSettledBad]) and [IsAmtNonNeg] and [IsQtyPositive] and
* [IsAcctLen10] and [IsCurrency3] and [IsTxnUnique] and [HasAccount] and [HasSecurity]
* **Transactions\_Exceptions**: filter DQ\_Flag\_Transactions = false, add Reason column (as you did for Accounts; include each failed flag).
* **Transactions\_Staging**: filter DQ\_Flag\_Transactions = true, remove helper columns.

**5) Corporate\_Actions – Landing & Pre-checks**

Duplicate into **Corporate\_Actions\_Pre**.

**A) Clean + Types**

* Action\_ID (Text), Security\_ID (Text), Action\_Type (Text), Action\_Date (Date), Amount (Decimal).
* Trim/Clean.

**B) Simple rules**

* **IsActionIdPresent**:
* not Text.IsNullOrEmpty([Action\_ID])
* **HasSecurity**: Merge with **Securities\_Staging** on Security\_ID, expand one column and check not null.
* **IsActionDatePlausible** (optional):
* ([Action\_Date] <= pToday) and ([Action\_Date] >= Date.AddYears(pToday,-3))

**C) Split outputs**

* **Corporate\_Actions\_Exceptions**: any failed flags.
* **Corporate\_Actions\_Staging**: passed rows only.

**6) Build a DQ Summary (nice interview touch)**

Create a new **DQ\_Summary** query using **Enter Data** with the rule names, then **Append Queries** from counts you compute per exceptions table:

* For each exceptions query (e.g., Accounts\_Exceptions), **Add Column → Custom** to compute RuleCount = Table.RowCount(#"PreviousStep"), then keep as a single-row summary (use Group By: **All Rows → Count Rows**).
* Append all summaries into one table that lists:
  + TableName, Rule/Reason, Violations, % of total.

(You can also just show **Column quality** view live during the demo.)

**7) Produce final “staging” outputs for SQL load**

Your **staging outputs** to load into SQL are:

* Accounts\_Staging
* Securities\_Staging
* Transactions\_Staging
* Corporate\_Actions\_Staging

And your **exceptions**:

* Accounts\_Exceptions
* Securities\_DuplicateIDs (if any)
* Transactions\_TypeErrors
* Transactions\_Exceptions
* Corporate\_Actions\_Exceptions

**Export options**

* **Power BI:** Home → Transform Data → right-click a table → **Copy Table** (paste to CSV), or use **DAX Studio**/**Tabular Editor** if you prefer.
* **Excel Power Query:** **Close & Load To… → Table (new worksheet)** for each staging/exceptions query, then **File → Save As → CSV** per sheet for bulk load to SQL.

Once exported, use SSMS **Import Flat File** or BULK INSERT to load to staging.\* tables.

**8) Acceptance criteria (ready to move to Step 2)**

* **Accounts\_Staging**: 0 dup PKs, all 10-digit accounts, valid ISO-like currency codes.
* **Securities\_Staging**: 0 dup PKs, all IDs present.
* **Transactions\_Staging**: 0 future trades; settled rows have valid settlement dates; non-negative amounts; FK-clean; unique Transaction\_ID.
* **Corporate\_Actions\_Staging**: FK-clean; plausible dates.
* **Exceptions** tables exist with clear Reasons for RCA.

Steps 1 & 2 done, here’s exactly what to do next (copy-paste friendly). I’ll assume **SQL Server + SSMS** and **Power BI Desktop**.

# 3) Load to SQL (staging schema)

**3.1 Create DB & schemas (T-SQL)**

3.2 Create **staging** tables (raw-after-PQ; no constraints)

**Load CSV → staging**  
In SSMS: **DB → Tasks → Import Flat File…** (use the four CSVs)  
Or via BULK INSERT (replace the file paths):

# 4) Post-ETL DQ enforcement → promote to prod (and log exceptions)

**4.1 Create prod tables (with constraints)**

**4.2 Exceptions + run-log tables**

**4.3 Promote Accounts (gate + exceptions)**

**4.4 Promote Securities (simple)**

**4.5 Promote Transactions with full rule gating + exceptions**

# 5) Build the Power BI model (live SQL + RLS)

**5.1 Connect & model**

* **Get Data → SQL Server → connect to CustodyDemo → select prod.Accounts, prod.Securities, prod.Transactions.**
* **Relationships:**
  + **prod.Transactions[Account\_Number] → prod.Accounts[Account\_Number] (Many-to-One, single direction)**
  + **prod.Transactions[Security\_ID] → prod.Securities[Security\_ID] (Many-to-One)**

**5.2 Core KPIs (DAX)**

**Total AUC :=**

**SUM ( 'prod.Transactions'[Trade\_Amount] )**

**Failed Trade % :=**

**DIVIDE (**

**CALCULATE ( COUNTROWS('prod.Transactions'), 'prod.Transactions'[Status] = "Failed" ),**

**COUNTROWS('prod.Transactions')**

**)**

**Avg Settlement Days :=**

**AVERAGEX (**

**FILTER ( 'prod.Transactions', NOT ISBLANK('prod.Transactions'[Settlement\_Date]) ),**

**DATEDIFF ( 'prod.Transactions'[Trade\_Date], 'prod.Transactions'[Settlement\_Date], DAY )**

**)**

**DQ Score (clean rows only) :=**

**VAR CleanRows =**

**COUNTROWS ( 'prod.Transactions' )**

**VAR AllRows =**

**CALCULATE ( COUNTROWS ( 'prod.Transactions' ), ALL ( 'prod.Transactions' ) )**

**RETURN DIVIDE ( CleanRows, AllRows )**

# 6) Lineage documentation (Business Level, Technical Level, Attribute level)