A. Gap Analysis & Mapping Current vs Future States

1. **Identify As-Is vs To-Be**
   – **As-Is:** How things work today (systems, data flows, manual steps, error rates).
   – **To-Be:** The target state after your solution (automated flows, new platforms, controls, KPIs).
2. **Create a Gap Matrix**

Capability/Process	As-Is	To-Be	Gap Description	Priority
Data Extraction	Manual CSV export via email	Scheduled ETL job into staging	Build scheduling & connection scripts	High
Validation Rules	Ad hoc Excel checks (15% errors)	Automated rule engine (<1% errors)	Develop transformation & QC scripts	High
Metadata Management	None	Central metadata repository	Select/configure metadata tool	Medium
Audit Logging	No audit trail	Full ETL execution log	Implement logging framework	Medium

3. – **Priority** helps you decide which gaps to tackle first.
4. **Derive Requirements**
   – For each “Must” gap, capture a BRD requirement.
   – Translate high-level BRD items into FRD field-level specs and transformation logic.

B. Mock BRD–FRD Pair #1: Loan-Booking Data Migration

1. As-Is vs To-Be Snapshot

Aspect	As-Is	To-Be	Gap	Priority
Source System	Legacy Access DB	Oracle Data Warehouse	Build ETL extract connector	High
Data Quality	~12% reconciliation errors (manual)	<0.5% automated validation	Create SQL-based QC rules	High
Process Scheduling	Manual nightly run via email reminder	Cron-scheduled ETL with alerting	Set up scheduler & monitoring	High
Metadata	No definitions	Data dictionary with business glossary	Populate dictionary from FRD mappings	Medium

---

1.1 Business Requirements Document (BRD)

ID	Requirement Description	Priority (MoSCoW)
BRD-01	Extract “LoanMaster” & “LoanTxn” tables nightly from Access into staging.	Must
BRD-02	Transform date formats to `YYYY-MM-DD`.	Must
BRD-03	Flag & report null `CustomerID` records.	Should
BRD-04	Log row counts & execution status for each ETL run.	Must
BRD-05	Provide metadata definitions for each target field in a central dictionary.	Should

---

1.2 Functional Requirements Document (FRD)

Source Table	Source Column	Target Table	Target Column	Data Type	Transformation Rule
LoanMaster	LoanID	dw.loan_master	loan_id	VARCHAR(20)	None
LoanMaster	OriginationDate	dw.loan_master	orig_date	DATE	`TO_DATE(OriginationDate, 'DD-MM-YYYY')`
LoanTxn	Amount	dw.loan_txn	txn_amount	NUMBER(12)	Cast and remove commas
LoanTxn	TransactionDate	dw.loan_txn	txn_date	DATE	`TO_DATE(TransactionDate,'DD-MM-YYYY HH24:MI:SS')`
LoanMaster	StatusCode	dw.loan_master	status_desc	VARCHAR(10)	Map “A”→“Active”, “C”→“Closed”, else “Unknown”

Business Rules & Validation

- **BR-R1:** Reject null `CustomerID`; write to `dw.err_log` with error code.
- **BR-R2:** If `txn_amount` ≤ 0, flag for manual review.
- **BR-R3:** Ensure unique `LoanID`; on duplicates, keep record with latest `LastUpdated`.

C. Mock BRD–FRD Pair #2: Trade Finance LC Data Consolidation

2. As-Is vs To-Be Snapshot

Aspect	As-Is	To-Be	Gap	Priority
Data Source	Multiple Excel templates from branches	Automated ingest from Core Banking API	Develop REST connector & mapping logic	High
Data Validation	Manual cross-sheet checks	Rule-based engine for format & completeness	Define & build validation rules	High
Reporting Latency	T+2 days for reconciled LC status report	T+0 (same day) via dashboard	Automate ETL + refresh of reporting layer	High
Lineage Tracking	Ad hoc logs	End-to-end lineage documented in metadata store	Capture lineage metadata in ETL framework	Medium

---

2.1 Business Requirements Document (BRD)

ID	Requirement Description	Priority
BRD-11	Ingest LC Issuance & Amendment data nightly via Core Banking API.	Must
BRD-12	Validate mandatory fields (LC Number, Beneficiary, ExpiryDate) for completeness.	Must
BRD-13	Provide error report for missing/invalid records.	Should
BRD-14	Load cleansed data into `dw.trade_lc` schema.	Must
BRD-15	Update metadata store with lineage: Source API → Staging → DW.	Could

---

2.2 Functional Requirements Document (FRD)

Source	Source Field	Target Table	Target Field	Type	Transformation / Validation Rule
API_LC_Issues	LCNumber	dw.trade_lc	lc_number	VARCHAR(20)	None
API_LC_Issues	BeneficiaryName	dw.trade_lc	beneficiary	VARCHAR(100)	Trim whitespace
API_LC_Issues	Amount	dw.trade_lc	lc_amount	NUMBER(15)	Convert to numeric
API_LC_Issues	IssueDate	dw.trade_lc	issue_date	DATE	`TO_DATE(IssueDate,'YYYY-MM-DD"T"HH24:MI:SS"Z"')`
API_LC_Issues	ExpiryDate	dw.trade_lc	expiry_date	DATE	Same as above

Business Rules & Validation

- **BR-R11:** Reject any LC with missing `LCNumber` or `ExpiryDate`.
- **BR-R12:** If `lc_amount` > 5M USD, flag for manual compliance check.
- **BR-R13:** Log all load errors to `dw.lc_err_log` with error type and timestamp.