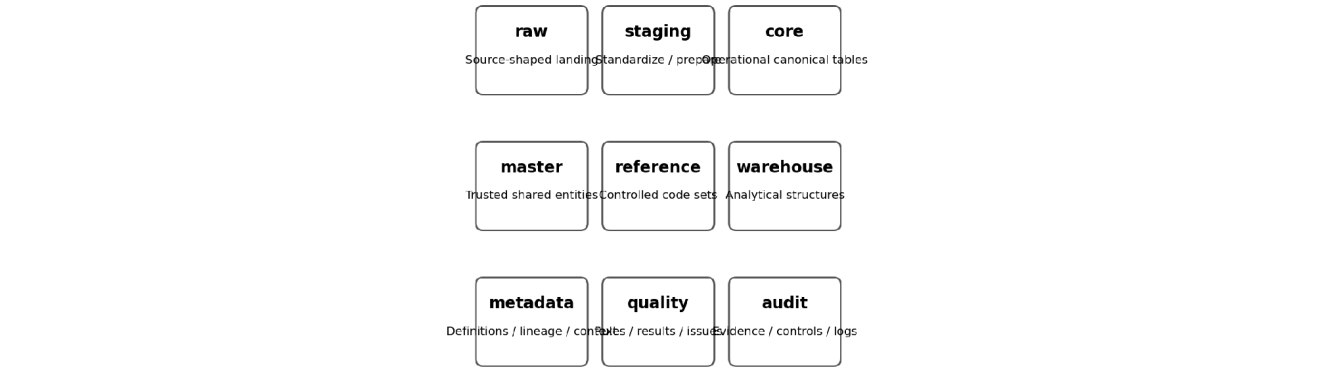
🧰 Meridian Lab Environment & Setup Blueprint

A reproducible Windows-based environment for CDMP applied learning

CDMP DMBOK Mastery Lab • Applied Learning Companion Program



These schema names are a Meridian lab design — not a DAMA-prescribed architecture. Their purpose is to make source, transformation, trust, analytics, metadata, quality and evidence visibly separable.

Logical training schemas. These are a Meridian lab design, not a DAMA-prescribed architecture.

1. Environment principles

- Local-first and understandable: avoid cloud complexity until a chapter truly benefits from it.
- Reproducible: another clean workstation or a future-you should be able to rebuild the lab from documented installs, SQL and Python requirements.
- Synthetic only: no real sensitive or employer data.
- Source-preserving: raw/source files remain unchanged; transformations create new staged/curated outputs.
- Script-first for learning-critical database state: important objects should be creatable from SQL, not only GUI clicks.
- Evidence-aware: outputs, screenshots and decision records have known locations.
- Destroyable: the lab can be reset without risking personal files or production systems.

2. Recommended base software

Software	Role	Initial status
PostgreSQL 18	Primary relational lab DBMS.	Required.
DBeaver Community	Database navigator/SQL workbench.	Required.
Python 3.x + venv	Scripting environment.	Required.
pandas	DataFrame profiling/transformation.	Required after Python foundations.
VS Code + Python extension	Code/project workspace.	Required.

Software	Role	Initial status
Git	Version control/evidence.	Required early.
PowerShell	Windows command-line literacy and setup.	Built into Windows; learn basics.
diagrams.net	ERD/architecture/lineage visuals.	Recommended.
DuckDB	Later file analytics.	Optional until triggered.
Metabase	Later BI/dashboard lab.	Optional until Chapter 11.
OpenMetadata + Docker	Later metadata/catalog/quality lab.	Optional advanced; do not install initially.

3. Local project structure

📁 **Suggested local repository:** meridian-cdmp-lab/

```
README.md
.gitignore
requirements.txt
docs/
data/
  raw/
  reference/
  generated/
sql/
  00_setup/
  01_core/
  02_security/
  03_integration/
  04_master_reference/
  05_warehouse/
  06_quality/
python/
  generators/
  profiling/
  integration/
  quality/
diagrams/
evidence/
backups/
```

4. PostgreSQL database and schema plan

Use one local training database named meridian_lab. Within it, schemas make different learning purposes visible. These names are intentionally pedagogical. When a chapter discusses architecture, schemas or metadata, distinguish the Meridian implementation choice from DAMA's conceptual guidance.

Schema	Meridian purpose	Typical contents
raw	Landing/source-shaped database inputs when SQL loading is useful.	Source-shaped CRM/POS extracts.
staging	Clean/standardize/prepare before trusted integration.	Parsed/normalized fields, source keys.
core	Operational canonical relational model for the lab.	Customer, Product, Order, OrderItem, Supplier, Location.
master	Trusted/mastered shared entities created in later labs.	Customer master, supplier/product mastered views/tables.
reference	Controlled code/value sets.	Country/state, order status, return reason, product category.
warehouse	Analytical structures.	Dimensions, facts, analytical views.
metadata	Lab-maintained technical/business context when stored relationally.	Definitions, source/target mapping, lineage records.
quality	Rules, profiling results and issue observations.	DQ rules, test results, issue history.
audit	Selected evidence/control logs for training.	Load/run records, validation outcomes.

5. Database roles for security labs

Role	Purpose
meridian_admin	Lab owner for setup only; use deliberately, not as everyday analyst.
meridian_loader	Loads approved data into designated schemas.
meridian_analyst	Reads approved analytical/curated data; should not automatically access all sensitive columns.
meridian_steward	Reads domain data and maintains selected reference/master stewardship artifacts where the lab requires.
meridian_bi	Reads warehouse views needed for dashboards.
meridian_restricted_test	Deliberately constrained account used to prove denied access.

6. Rebuild philosophy

1. Install software from official sources and record versions.
2. Clone/copy the lab repository.
3. Create a fresh Python virtual environment and install declared packages.
4. Create meridian_lab and required roles using controlled setup steps.
5. Run schema/DDL scripts in order.
6. Generate or copy synthetic source files.
7. Load data using documented SQL/Python.
8. Run validation checks and compare expected counts/constraints.

9. Only then begin a chapter lab.

7. Safety and rollback rules

- Never point lab scripts at a work or production database.
- Use a recognizable lab database name and local connection profile.
- Before destructive UPDATE/DELETE exercises: SELECT the target first; use a transaction when practical; record expected row count.
- Do not store passwords in source code or Git. Use local environment/config practices taught in the academy.
- Keep a known-good rebuild path so destructive “break it” labs are safe.
- Preserve raw source files; create new outputs instead of overwriting evidence-producing inputs.