🧭 TECHNOLOGY COMPETENCY ROADMAP — WHAT TO LEARN, WHY IT MATTERS, AND WHERE IT FITS

CDMP Applied Data Management Laboratory • Meridian Commerce Group

🎯 PURPOSE

This document is the permanent technology roadmap for the CDMP Applied Data Management Laboratory. It is not a ChatGPT-authored technology course and it is not intended to replace the learning resources you choose for yourself. Its job is to tell you exactly what technologies and software are worth learning, why each one supports Data Management, how each one will be used in the Meridian Commerce Group labs, how deeply you need to learn it, what level of independent performance proves you are ready, and where to stop so technology does not take over the CDMP goal.

The governing rule is simple:
DMBOK THEORY → TECHNOLOGY SKILL → MERIDIAN BUSINESS PROBLEM → HANDS-ON WORK → EVIDENCE → DMBOK DEBRIEF.

Technology is never the lesson by itself. The DMBOK concept is the lesson. Technology gives you a way to see, test, build, break, repair, trace, measure or explain that concept.

1. HOW TO USE THIS ROADMAP

• Choose your own courses, books, videos, tutorials or documentation for each technology.
• You do not need to finish every technology before beginning the CDMP Applied Lab program. Learn just ahead of need.
• Do not enter a lab with a major tool skill treated as unexplained magic. If a lab requires a skill you cannot explain, pause the lab and train that skill first.
• Learn to the target depth shown here. Do not keep going simply because a course contains more advanced material.
• Use the Technology Skills tab in the CDMP Applied Lab & Technology Mastery Tracker to record your progress and evidence.
• A supplied lab script is scaffolding, not mastery. You should eventually be able to explain the learning-critical logic, change it, predict the result and rebuild a smaller equivalent independently.

2. THE TECHNOLOGY STACK AT A GLANCE

🔵 CORE TECHNICAL SPINE

SQL — strongest priority; query, inspect, validate, reconcile and transform structured data.
PostgreSQL — real relational database environment in which models, keys, constraints, roles, views and warehouse structures exist.
DBeaver Community — visual database workbench used to inspect PostgreSQL objects, metadata, relationships, SQL and results.
Python — general scripting language for repeatable file/data work, automation, validation and integration.
pandas — Python library that makes tabular profiling, cleaning, reconciliation, merging and validation practical.

🟢 CORE SUPPORTING TOOLS

VS Code — working environment for SQL, Python, Markdown, project files and Git.
PowerShell / Windows command line — practical environment, path, file, executable and virtual-environment literacy.
Git — version history, reproducibility, change evidence and controlled development of lab artifacts.
CSV / JSON / Parquet literacy — understand the formats through which data arrives, moves and is analyzed.
diagrams.net — architecture, ERD, lineage, lifecycle and flow diagrams.
Google Drive / Docs / Sheets — management artifacts, inventories, registers, decision records, evidence and governance work.

🟠 LATER, CHAPTER-TRIGGERED TOOLS

REST APIs + Postman — integration and interoperability practice when systems exchange request/response payloads.
Metabase — turn warehouse data into business questions, KPIs and dashboards for DW/BI work.
OpenMetadata — make catalog, glossary, ownership, classification, lineage and quality concepts visible after they are understood manually.
DuckDB — lightweight analytical SQL directly against files such as CSV and Parquet.
Docker / Docker Compose — only if needed to run later tools such as OpenMetadata locally; not a general container-engineering goal.

3. RECOMMENDED LEARNING SEQUENCE

STAGE A — ENVIRONMENT + RELATIONAL FOUNDATION

1. VS Code and basic PowerShell/file-system literacy.
2. Install PostgreSQL and DBeaver so you have a real place to practice.
3. SQL Foundations I: reading/filtering/sorting data and understanding NULL and grain.
4. SQL Foundations II: joins, grouping, aggregation, CTEs and relational reasoning.
5. PostgreSQL object fundamentals: schemas, tables, keys, constraints, views, transactions and roles.
6. DBeaver workbench fluency: browse, inspect, query and interpret an unfamiliar schema.

STAGE B — DATA MANAGEMENT DIAGNOSTICS

7. SQL for Data Management: profiling, duplicate detection, referential checks, reconciliation, quality rules and safe data changes.
8. Data format literacy: CSV, JSON, dates/timestamps, identifiers and Parquet awareness.
9. Git basics for versioned, reproducible work.

STAGE C — PROGRAMMATIC DATA WORK

10. Python foundations.
11. pandas for Data Management.
12. Python ↔ PostgreSQL connection concepts and repeatable load/validation workflows.

STAGE D — CHAPTER-TRIGGERED TOOLS

13. Postman/API literacy when Chapter 8 integration work needs it.
14. Metabase when Chapter 11 DW/BI is ready.
15. OpenMetadata when Chapter 12/13 metadata and quality concepts are already understood manually.
16. DuckDB and deeper Parquet work when Chapter 14 creates a legitimate scale/format need.

4. CORE TECHNOLOGY — SQL

🎯 Why you are learning it

SQL is the most important technical skill in this program because many Data Management concepts concern what is actually present in structured data, how different datasets relate, whether rules are being followed, whether systems agree, and whether information can be reconciled. SQL lets you ask those questions directly instead of accepting screenshots or reports produced by someone else.

🧩 How it fits the CDMP program

SQL makes Data Modeling, Storage, Security, Integration, Master/Reference Data, DW/BI, Metadata and Data Quality concrete. In Meridian you will use SQL to inspect schemas, find duplicate customers, check missing values, test invalid references, reconcile source populations, compare reference codes, trace transformations, create trusted views, calculate measures and validate repairs.

📚 What to learn

• SELECT, FROM, WHERE, ORDER BY, aliases, DISTINCT and LIMIT.
• Data types, NULL logic, comparisons, Boolean logic and CASE.
• Primary-key and foreign-key concepts.
• INNER JOIN and LEFT JOIN first; other joins at practical recognition depth.
• Query grain: what one output row represents and how joins change it.
• COUNT, SUM, AVG, MIN, MAX, GROUP BY and HAVING.
• Subqueries and common table expressions (CTEs).
• Set operations such as UNION and practical population comparison.
• String/date cleanup and conversion functions at practical depth.
• Window functions at useful analytical/diagnostic depth.
• INSERT, UPDATE and DELETE with safe before/after validation.
• Transactions and rollback when changing data.
• Profiling queries, duplicate detection, missing/invalid-value checks, referential-integrity checks and reconciliation queries.

✅ Ready when

Given a business question, you can identify the required tables and expected output grain, write the query from a blank editor, explain why the join/aggregation is correct, interpret the result, and detect when the result is suspicious because row counts or grain changed unexpectedly.

🛑 Stop before

Advanced performance tuning, complex procedural SQL, vendor-specific expert features and database-programming specialization. Those are useful careers in their own right but are not required for the purpose of this CDMP lab program.

5. CORE TECHNOLOGY — POSTGRESQL

🎯 Why you are learning it
SQL is a language; PostgreSQL gives that language a real database environment. It lets abstract DMBOK ideas such as schema, table, relationship, key, constraint, view, role, transaction and warehouse structure become actual objects you can inspect and manipulate.

🧩 How it fits the CDMP program
PostgreSQL is the primary Meridian lab database. It becomes especially important in Chapters 5–8 and 10–13. You will implement models, enforce integrity, practice storage/operations, create access roles, stage integrated data, build master/reference structures, create warehouse tables, inspect technical metadata and store quality results.

📚 What to learn
• Server/database/schema/table/view distinctions.
• Creating and connecting to a local training database.
• CREATE TABLE and practical PostgreSQL data types.
• Primary keys, foreign keys, NOT NULL, UNIQUE and CHECK constraints.
• Identity/surrogate-key awareness.
• Schemas and qualified object names.
• Views and when they are useful.
• BEGIN, COMMIT and ROLLBACK.
• Users/roles, GRANT and REVOKE.
• Basic index purpose and inspection.
• Metadata/catalog inspection at awareness/practical level.
• Import/export patterns useful to the lab.
• Backup/rebuild/dump-restore awareness sufficient to safely restore the local training environment.

✅ Ready when
You can connect to the Meridian lab database, create a schema/table from SQL, define useful constraints, inspect the object afterward, use a transaction safely, create a simple role/permission example and explain what each object is doing.

🛑 Stop before
Production database clustering, high-availability engineering, deep query tuning, replication administration, operating-system database administration and PostgreSQL internals. The goal is Data Management fluency, not becoming a production DBA.

6. CORE TECHNOLOGY — DBEAVER COMMUNITY

🎯 Why you are learning it
A database can remain mentally invisible if all you ever see is a command prompt. DBeaver lets you visually explore a relational environment while still requiring you to understand SQL and database objects. It is your main database workbench, not a substitute for SQL.

🧩 How it fits the CDMP program
DBeaver supports Chapters 4–13 by allowing you to inspect Meridian schemas, columns, data types, keys, constraints, views, DDL, relationships, result sets and technical metadata. It is especially useful for Data Architecture, Modeling, Storage, Security and Metadata because you can see the structures you are reasoning about.

📚 What to learn
• Create and identify a database connection; understand host, port, database and user.
• Database Navigator: databases, schemas, tables, views and other objects.
• Object properties: columns, keys, constraints, indexes and metadata.
• SQL Editor: create, save, reopen and execute scripts intentionally.
• Difference between running a selected statement and a full script.
• Result grids and safe export/import of lab data.
• View or generate DDL when useful.
• ERD/relationship navigation as support for learning.

✅ Ready when
You can open an unfamiliar Meridian schema, locate the important objects, explain their basic structure before querying, find the table DDL/metadata, run and save SQL intentionally and export a result without being led click-by-click.

🛑 Stop before
Memorizing every DBeaver feature or performing work through GUI clicks you cannot explain in SQL/database terms.

7. CORE TECHNOLOGY — PYTHON

🎯 Why you are learning it
Not every Data Management problem lives inside one relational database. Data arrives as files, JSON payloads, extracts and repeated processes. Python gives you a reusable way to inspect, generate, validate and move those data assets without turning every task into manual spreadsheet work.

🧩 How it fits the CDMP program
Python becomes most important in Chapters 8, 10, 13 and 14. Meridian will use it for synthetic-data generation, reading source files, parsing JSON, repeatable validation, transformations, controlled defect injection, file inspection and automation of work that would be tedious or fragile by hand.

📚 What to learn
• Variables and fundamental data types.
• Lists, dictionaries, sets and tuples.
• Conditions and loops.
• Functions and reusable logic.
• Modules/imports.
• pathlib and safe file/folder handling.
• Reading/writing text, CSV and JSON.
• Exceptions and useful error messages.
• Python virtual environments and package installation concepts.
• Simple command-line scripts.
• Basic database connection concepts when later integrating Python and PostgreSQL.

✅ Ready when
You can start with a plain-language requirement, create a small script from a blank file, read an input, apply repeatable logic, write an output, handle a predictable failure and explain each learning-critical step without copying a finished template.

🛑 Stop before
Web frameworks, application development, algorithm-heavy computer science, advanced object-oriented architecture and software-engineering specialization unless another career goal later requires them.

8. CORE TECHNOLOGY — PANDAS

🎯 Why you are learning it
pandas is what makes Python directly useful for much of the tabular work in Data Management. It allows you to inspect, compare, profile, normalize, merge and validate datasets while preserving repeatability.

🧩 How it fits the CDMP program
pandas is particularly valuable in Chapters 8, 10, 13 and 14. In Meridian you will compare customer sources, normalize differing representations, identify candidate duplicates, inspect missing values, reconcile counts, parse file data, create exception datasets and prepare outputs for PostgreSQL or analytical work.

📚 What to learn
• DataFrame and Series mental model; rows, columns, index and dtypes.
• read_csv, read_json, read_sql and read_parquet; corresponding write methods where appropriate.
• head, tail, info, dtypes, value_counts and describe.
• Selecting/filtering rows and columns.
• Missing-value identification and business-aware handling.
• Exact and candidate duplicate detection.
• String normalization, type conversion and date/timestamp parsing.
• merge/join with explicit keys and expected cardinality.
• groupby and aggregations.
• Validation/rule checks and exception datasets.
• Writing curated outputs while keeping raw/source inputs unchanged.

✅ Ready when
You can load an unfamiliar Meridian file, explain its shape and grain, identify obvious quality concerns, merge it with another dataset while validating the row-count effect, produce a clean/exception output and explain the DataFrame state before and after each transformation.

🛑 Stop before
Advanced scientific/numerical computing, machine learning and specialized data-science workflows. Those are not the reason pandas is in this program.

9. SUPPORTING TECHNOLOGY — VS CODE

🎯 Why you are learning it
VS Code gives the Meridian project one controlled workspace for SQL, Python, Markdown, configuration notes and Git rather than scattering scripts and evidence across unrelated folders.

🧩 Program fit

Used throughout the technical labs for editing SQL/Python, reading project documentation, selecting the correct Python environment and keeping the local repository understandable.

📚 Learn
Project/workspace navigation, opening folders, file extensions, integrated terminal, Python interpreter selection, running/debugging simple scripts, search, basic extensions and source-control view awareness.

✅ Ready when
You can open the Meridian repository, find the correct script/data folder, select the project Python environment, run a script and understand which working directory and files are involved.

🛑 Stop before
Turning editor customization into a separate learning project.

10. SUPPORTING TECHNOLOGY — POWERSHELL / WINDOWS COMMAND LINE

🎯 Why you are learning it
Many technical problems that appear to be “Python problems” or “database problems” are actually path, folder, environment, executable or working-directory problems. Basic command-line literacy helps you understand what your computer is doing.

🧩 Program fit
Install/setup validation, navigating Meridian folders, running commands, checking versions, activating Python environments, executing scripts and rebuilding the local lab.

📚 Learn
pwd/Get-Location, cd/Set-Location, dir/Get-ChildItem, paths, absolute vs relative paths, file creation/copy/move awareness, environment variables, PATH concept, running executables, basic piping/output awareness and virtual-environment activation.

✅ Ready when
You can navigate to the right project directory, identify where a file/program actually lives, run a documented command intentionally and recover from being in the wrong folder.

🛑 Stop before
PowerShell automation specialization. Practical literacy is enough.

11. SUPPORTING TECHNOLOGY — GIT

🎯 Why you are learning it
Git gives you controlled history. That makes concepts such as versioning, traceability, reproducibility and change evidence tangible in your own lab work.

🧩 Program fit
Use Git for SQL, Python, README files, definitions, quality rules, mappings and important lab documents. It lets you compare before/after logic and reconstruct how a Meridian artifact changed.

📚 Learn
Repository concept, init/clone awareness, status, add, commit, log, diff, .gitignore, basic branches/merge, meaningful commit messages and the rule to never store secrets or sensitive material.

✅ Ready when
You can create a meaningful lab milestone, inspect what changed, compare versions and recover the reason a script/document evolved by reading its history.

🛑 Stop before
Enterprise branching strategies, release engineering and CI/CD unless a later career goal requires them.

12. SUPPORTING KNOWLEDGE — CSV, JSON AND PARQUET

🎯 Why you are learning them
The same business fact can be represented differently by different systems. Understanding file formats makes integration, metadata and data-quality problems easier to recognize.

CSV — flat tabular interchange. Learn delimiters, quoting, headers, encoding, blanks vs NULL, dates and schema assumptions.
JSON — hierarchical/semi-structured data. Learn objects, arrays, nesting, keys, values, nulls and schema/version change awareness.
Parquet — column-oriented analytical file format. Learn why it differs from CSV conceptually, why schema/types matter and why it is useful for analytical workloads.

🧩 Program fit

Chapter 8 uses CSV/JSON to demonstrate integration and schema differences. Chapter 12 uses the formats to reinforce technical metadata. Chapter 13 uses representation defects for quality work. Chapter 14 uses JSON/Parquet to introduce larger analytical/event data without building a distributed-computing platform.

✅ Ready when
You can inspect an unfamiliar file, identify its structural assumptions, explain what metadata would be needed to interpret it correctly and diagnose a basic format/schema problem.

13. SUPPORTING SOFTWARE — DIAGRAMS.NET

🎯 Why you are learning it
Not all Data Management knowledge is best expressed in code. Architecture, models, lineage, lifecycles, roles and information flows are relational concepts that are often clearer visually.

🧩 Program fit
Chapter 1 lifecycle/enterprise views, Chapter 4 architecture, Chapter 5 models/ERDs, Chapter 8 source-to-target/integration flows, Chapter 12 lineage and Chapter 17 change/process flows.

📚 Learn
Basic shapes/connectors, labels, grouping, swimlane awareness, ERD notation support, export to PNG/PDF and disciplined arrow semantics.

✅ Ready when
You can create a diagram in which every object and arrow has a defined meaning and another learner could explain the flow without relying on decorative assumptions.

14. MANAGEMENT ARTIFACT TOOLS — GOOGLE DRIVE, DOCS AND SHEETS

🎯 Why they remain part of the technology environment
Several DMBOK chapters are about governance, ownership, policy, ethics, maturity, content, roles and organizational change. Forcing those chapters into code would distort the learning. Documents, registers and decision records are the correct medium for much of that work.

🧩 Program fit

Drive/Docs/Sheets support data inventories, governance charters, RACI/responsibility matrices, ethical decision records, issue logs, glossary work, content/retention registers, maturity evidence, change plans and the master lab tracker.

Target depth
Comfortable professional use: structured documents, tables, filters, simple formulas, organized Drive folders, hyperlinks and evidence tracking. This is not intended to become separate Google Workspace certification study.

15. LATER TOOL — REST APIS + POSTMAN

🎯 Why you will learn it
Modern systems frequently exchange data through APIs rather than files or direct database connections. API literacy makes Data Integration & Interoperability more realistic.

🧩 Program fit
Primarily Chapter 8. You can observe a request sent to a service, inspect the JSON response, reason about fields and identifiers, see status/error behavior and understand how schema/version changes affect downstream consumers.

📚 Learn
Endpoint concept, request/response, common HTTP methods at practical depth, headers, status codes, query parameters, JSON payloads, authentication conceptually, pagination and version/schema awareness. Use Postman to send and inspect requests without writing a full application.

✅ Ready when
You can explain what data went into a request, what came back, what a failure/status means and how the response could be mapped into another system.

🛑 Stop before
Becoming an API developer, building complex services or learning full backend engineering.

16. LATER TOOL — METABASE

🎯 Why you will learn it
A warehouse exists to support information use. Metabase gives Chapter 11 a visible business-consumption layer so you can see how facts, dimensions, measures and definitions become dashboards.

🧩 Program fit
Chapter 11 Data Warehousing & BI and later metadata/lineage incidents. Meridian can create a revenue or customer KPI, then trace it backward from dashboard → warehouse → transformation → source.

📚 Learn
Connect to the lab database, ask simple questions, use SQL where useful, create a dashboard, understand filters/aggregation and inspect the data behind a visual.

✅ Ready when
You can build a simple dashboard and explain exactly which warehouse objects/definitions produce each metric.

🛑 Stop before
BI product administration or visualization-specialist training.

17. LATER TOOL — OPENMETADATA

🎯 Why you will learn it
OpenMetadata makes metadata-management ideas tangible inside an actual catalog-style product: assets, descriptions, glossary terms, owners, classifications, lineage and quality context.

🧩 Program fit
Primarily Chapters 12 and 13, optionally Chapter 3 after governance concepts are learned manually. The rule is concept first, product second. First understand glossary, dictionary, catalog, ownership, lineage and classification in DAMA terms; then see how a product operationalizes those ideas.

📚 Learn
Browse/catalog assets, add descriptions/owners, create/use glossary terms, inspect lineage, apply classifications and observe quality/governance features at practical depth.

✅ Ready when
You can take a real Meridian data asset and use the catalog to answer: What is this? Where is it? Who is responsible? Where did it come from? What does it mean? What controls/context apply?

🛑 Stop before
OpenMetadata platform administration, production deployment architecture and product certification.

18. LATER TOOL — DUCKDB

🎯 Why you will learn it
DuckDB lets you run analytical SQL directly against files. It provides a simple way to contrast operational relational work with file-oriented analytical work without introducing a full distributed data platform.

🧩 Program fit
Primarily Chapter 14 and any earlier lightweight analytical-file exercise where it adds real learning value. Meridian can query larger click/event files in CSV/Parquet and compare that workflow with PostgreSQL operational data.

📚 Learn
Open/query CSV and Parquet, basic SQL in DuckDB, inspect schema, aggregate data and understand when direct analytical file querying is useful.

✅ Ready when
You can query an analytical file directly, explain its schema/grain and compare why you used DuckDB/file analytics instead of treating the task as an operational PostgreSQL workload.

19. LATER SUPPORT — DOCKER / DOCKER COMPOSE

🎯 Why it may appear
Some later learning platforms are easiest to run in containers. Docker is therefore an environment-enablement skill, not a primary Data Management subject.

🧩 Program fit
Potentially OpenMetadata and other optional later tools. Learn only enough to start/stop the required local stack, understand containers/images/ports/volumes/configuration at a basic level and safely reset the learning environment.

🛑 Stop before
Container orchestration, Kubernetes, production container engineering and cloud platform operations.

20. PYTHON ↔ POSTGRESQL CONNECTION SKILL

This is not a separate career track, but it is an important bridge skill after Python/pandas and PostgreSQL are comfortable.

📚 Learn

Database connection concepts, drivers/SQLAlchemy awareness, safe credential handling, parameterized queries, reading SQL results into pandas, writing controlled outputs back to PostgreSQL and transaction awareness.

🧩 Program fit
Chapters 8, 10, 11 and 13 where a repeatable file → Python → PostgreSQL → validation workflow makes the Data Management process visible.

✅ Ready when
You can move a synthetic Meridian dataset through Python into PostgreSQL, validate row counts/keys and explain what happened at each boundary.

CAREER DATA GOVERNANCE PLATFORM SPECIALIZATION — GOVERNANCE, NOT ENGINEERING

This layer is career-facing. It does not replace the vendor-neutral Meridian lab stack or the DMBOK. Learn the governance concept first, then learn how an enterprise platform operationalizes it.

Primary career platform — Microsoft Purview Data Governance

Use Microsoft Purview as the primary enterprise platform specialization after the relevant DMBOK concepts are understood. Keep the scope centered on Purview Data Governance: Data Map and Unified Catalog.

Learn deeply:
• governance domains and domain roles;
• data products and critical data;
• business concepts / glossary terms;
• owners, stewards, and accountability;
• metadata discovery, curation, and source-onboarding concepts;
• classification at governance-use depth;
• lineage interpretation and stewardship;
• Data Quality rules, scores, and governance-health evidence;
• governed discovery and right-use/access-policy concepts;
• translation from DAMA/DMBOK governance decisions into platform configuration.

Stop before the work becomes production ingestion/pipeline engineering, Fabric lakehouse or warehouse engineering, Azure Data Factory/Synapse/Databricks specialization, Spark

development, infrastructure-as-code, extensive SDK/API development, or deep Microsoft 365 security operations. Those may support governance environments, but they are not the professional identity this program is building.

Secondary platform — Collibra

Learn Collibra as the strongest secondary governance-specialist platform: stewardship, communities/domains, business glossary, ownership, policies, workflows, metadata, classification, and lineage. Build strong conceptual/workflow portability; pursue deeper hands-on work when access or a target employer justifies it.

Meridian practice sandbox — OpenMetadata

Keep OpenMetadata as the optional local catalog/lineage practice environment. It remains useful because it lets you practice catalog, glossary, ownership, lineage, classification, and quality concepts without making enterprise-license access a prerequisite for learning.

Employer-specific alternatives

Maintain recognition-level literacy in Informatica Cloud Data Governance and Catalog, Alation, and Atlan. Deepen one only when a target employer or role makes it relevant. Do not try to master every governance platform.

Certification guardrails

• CDMP Fundamentals remains the vendor-neutral core.
• DP-900 Azure Data Fundamentals is a recommended Microsoft foundation for a Purview path, not a Data Engineering commitment.
• Collibra Data Steward is an optional high-alignment platform credential if current access/eligibility permits.
• Informatica CDGC Foundation is optional and employer-driven; CDGC Professional is later and role-specific.
• SC-900 is awareness-only.
• SC-401 is not core Data Governance; use it only if a target role expands into information protection/security.
• Do not add SC-400; it is retired.
• Do not add DP-600 as a governance requirement; it is an analytics-engineering credential.
• Do not pursue a vendor SQL certification solely to prove governance ability; demonstrate governance-oriented SQL competence through evidence.

When this enters the study sequence

After Chapter 3: vendor-landscape awareness only. Recognize what enterprise governance/catalog platforms operationalize; do not memorize interfaces yet.
After Chapter 12: begin hands-on Microsoft Purview Data Governance specialization because glossary, metadata, ownership, catalog, and lineage concepts now have a DMBOK foundation.
During/after Chapter 13: extend the platform work into Data Quality rules, scores, stewardship, and governance-health evidence.
Portability proof: take one Meridian governance design, map it into Purview terminology/configuration, then explain how the same governance decision would transfer conceptually to Collibra or an employer-specific alternative.

Career-platform readiness

Do not mark a platform “learned” because a course was completed. Readiness means you can take a Meridian governance problem, state the business/governance requirement first, choose the relevant platform objects and roles, explain the resulting evidence, and translate every configuration back into DAMA/DMBOK language without drifting into engineering.

FINAL CAREER PLATFORM RULE: LEARN DATA GOVERNANCE FIRST. LEARN THE PLATFORM SECOND.

21. CHAPTER-TO-TECHNOLOGY CROSSWALK

Chapter 1 — Data Management
Primary medium: Docs/Sheets/diagrams.net + light SQL/DBeaver observation.
Why: inventory assets, identify value/risk, map lifecycle and separate enterprise Data Management from technology operation.

Chapter 2 — Data Handling Ethics
Primary medium: Docs/Sheets; SQL only if needed to inspect what data actually exists.
Why: the main skill is ethical judgment, not coding.

Chapter 3 — Data Governance
Primary medium: Docs/Sheets/diagrams + database metadata observation.
Why: build decision rights, stewardship, governance artifacts and resolve definition/authority disputes.

Chapter 4 — Data Architecture
Primary medium: diagrams.net + DBeaver + SQL metadata.
Why: map systems, stores and flows using a database/environment you can actually inspect.

Chapter 5 — Data Modeling and Design
Primary medium: SQL + PostgreSQL + DBeaver + diagrams.net.
Why: turn conceptual/logical/physical models into actual tables, relationships, keys and constraints.

Chapter 6 — Data Storage and Operations
Primary medium: PostgreSQL + DBeaver + SQL + PowerShell.
Why: operate, validate, back up/rebuild and observe how database state is maintained.

Chapter 7 — Data Security
Primary medium: PostgreSQL roles/views + SQL + DBeaver.
Why: classify data, grant/revoke access and prove both allowed and denied behavior.

Chapter 8 — Data Integration and Interoperability
Primary medium: SQL + PostgreSQL + Python/pandas + CSV/JSON + optional Postman.
Why: ingest different source shapes, stage/transform/reconcile them and trace source-to-target movement.

Chapter 9 — Document and Content Management
Primary medium: Drive/Docs/Sheets + file/metadata awareness; Python optional.
Why: content lifecycle, metadata, retention, version and discovery decisions are better practiced as content-management work than forced coding.

Chapter 10 — Reference and Master Data
Primary medium: SQL + PostgreSQL + Python/pandas.
Why: match conflicting customer sources, make survivorship decisions, build trusted outputs and control reference code sets.

Chapter 11 — Data Warehousing and Business Intelligence
Primary medium: PostgreSQL + SQL + Metabase + diagrams.
Why: build facts/dimensions, load analytical structures, define measures and trace dashboard KPIs to source.

Chapter 12 — Metadata Management
Primary medium: DBeaver/SQL metadata + Sheets/Docs + later OpenMetadata.
Why: catalog real Meridian assets, define business/technical metadata and trace lineage/ownership.

Chapter 13 — Data Quality Management
Primary medium: SQL + PostgreSQL + Python/pandas + optional OpenMetadata.

Why: inject defects, profile, define rules, measure quality, identify root cause, repair and show before/after evidence.

Chapter 14 — Big Data and Data Science
Primary medium: Python/pandas + DuckDB + JSON/Parquet.
Why: experience larger event data and different formats without turning the program into data-science or distributed-engineering training.

Chapter 15 — Data Management Maturity Assessment
Primary medium: Sheets/Docs + accumulated Git/evidence repository.
Why: use actual evidence from earlier labs to assess capability rather than scoring from impressions.

Chapter 16 — Data Management Organization and Role
Primary medium: Docs/Sheets + accumulated lab evidence.
Why: assign real activities already performed to owners, stewards, architects, technical roles and decision bodies.

Chapter 17 — Data Management and Organizational Change
Primary medium: Docs/Sheets/diagrams.
Why: practice stakeholder adoption, resistance, communication, training and transition of Data Management changes.

22. WHAT IS NOT REQUIRED FOR THIS PROGRAM

Do not add these simply because they appear in generic data-engineering roadmaps:
• Advanced Linux administration.
• Cloud architecture/certification solely for these labs.
• Kubernetes.
• Spark/Hadoop distributed engineering.
• Production data-orchestration platforms.
• Advanced DBA performance engineering.
• Machine learning/model development.
• Data-science mathematics.
• Advanced software engineering/OOP/algorithms.
• Enterprise DevOps/CI-CD.
• Multiple BI platforms.
• Multiple relational databases at the same time.

• dbt as a prerequisite. It can be considered later if it adds distinct learning value to warehouse transformation/testing/lineage, but SQL should be understood first and dbt is not needed to make the DMBOK program work.

23. HOW TO CHOOSE YOUR OWN TRAINING RESOURCES

When evaluating a course, book or tutorial, prefer resources that:
• teach from beginner foundations through the target depth listed here;
• make you type/build things yourself rather than only watch demonstrations;
• include exercises without giving the answer before you attempt them;
• teach why commands work, not only which buttons to press;
• use realistic data rather than isolated syntax fragments;
• include troubleshooting and interpretation of errors/results;
• let you practice from a blank editor;
• do not require you to finish advanced sections that are beyond the stop boundary in this roadmap.

A course does not have to mention DAMA or CDMP. Its job is to teach the technology. The Meridian labs will provide the Data Management context and connect the technology back to DMBOK concepts.

24. TECHNOLOGY READINESS GATES

Gate A — SQL Reader
You can write single-table filters/sorts, explain NULL, identify grain and interpret results.

Gate B — Relational SQL
You can join three tables correctly, aggregate without accidental duplication and use a CTE.

Gate C — PostgreSQL + DBeaver
You can connect, create a schema/table from SQL, inspect metadata, use constraints, save/run scripts and use a transaction safely.

Gate D — Python + pandas
You can create/activate a virtual environment, read CSV/JSON, inspect a DataFrame, filter/merge/group, detect nulls/duplicates and write an output.

Gate E — Reproducibility
You can run a project from documented steps, use basic Git status/commit/diff, keep raw inputs unchanged and preserve inspectable evidence.

Gate F — Applied Independence
When a Meridian lab gives you a problem instead of a command, you can choose the appropriate technology, explain why it fits, perform the learning-critical work with decreasing reliance on instructions and translate the result back into DMBOK language.

✅ FINAL RULE

You are not learning SQL, PostgreSQL, DBeaver, Python or any other tool because the CDMP exam is a software exam. It is not. You are learning them because they let you experience the structures, failures, decisions, movement, controls, evidence and consequences that the DMBOK describes. The technology makes the theory observable. The DMBOK debrief turns the observable experience back into exam-ready Data Management understanding.