JUPITER PRIME

Consulting • Training • Advisory

جوبيتر برايم للاستشارات والتدريب

COURSE BROCHURE

Advanced Data Warehouse and Lakehouse Engineering Training Course

22 November – 3 December 2026
Istanbul

jupiterprime.uk

info@jupiterprime.uk

Registered in the United Kingdom and Saudi Arabia

Advanced Data Warehouse and Lakehouse Engineering Training Course

Ref: 2560_97387 **Date:** 22 November – 3 December 2026 **Location:** Istanbul **Fees:** 9000 GBP

Data Warehouse and Lakehouse Engineering is a 2-week training course for data engineers and technical analysts who design the storage layer behind enterprise analytics, and it ends with a Storage Architecture Decision Record. As analytical estates grow across open table formats, object storage and governed schemas, organisations need storage designs that stay trusted, affordable and easy to query. Jupiter Prime Training offers this Data Warehouse and Lakehouse Engineering course for advanced practitioners.

Why Data Warehouse and Lakehouse Engineering Matters Now

Three forces shape storage design today. Open table formats now bring transactions, schema evolution and time travel to cheap object storage, blurring the old line between lake and warehouse. Finance leaders scrutinise consumption-based compute bills, so every partitioning and clustering choice carries a visible cost. Business owners and auditors expect conformed definitions and traceable history for every figure they sign off. Poor design means duplicated tables, conflicting metrics and slow queries. The course rests on dimensional modelling, data vault modelling and the medallion layering pattern.

Who Gains Most from Data Warehouse and Lakehouse Engineering

- Engineers responsible for physical table design, partitioning and file layout on warehouse and lakehouse platforms
- Analysts who own semantic definitions, conformed dimensions and the star schemas that feed dashboards
- Architects who choose between warehouse, lake and lakehouse patterns for new analytical domains
- Platform owners accountable for storage health, access policies and compute consumption
- Modellers who integrate history from many source systems into auditable hubs, links and satellites

A platform group that sends its architects, modellers and platform owners together gains most.

Data Warehouse and Lakehouse Engineering Capabilities

Participants move from comparing storage paradigms to defending a target design, with modelling, contracts and spend tied to that design.

- Assess warehouse, lake and lakehouse patterns against workload and latency needs — platform investment decisions
- Design star schemas with conformed dimensions and slowly changing history — business owners of key metrics
- Build data vault structures that absorb source change without remodelling — integration architects
- Specify data contracts covering schema, semantics and freshness expectations — producing and consuming domains
- Apply table layout, partitioning and compaction choices to query cost — finance and platform owners
- Communicate storage trade-offs in a written architecture decision — technology steering groups

Key Data Warehouse and Lakehouse Engineering Terms

- Lakehouse — a storage architecture that adds warehouse features such as transactions and schema enforcement to open files held in object storage.
- Open table format — a metadata layer over data files that records snapshots, schema and partitions so engines can read tables consistently.
- Conformed dimension — a dimension table shared by several fact tables with identical keys and attributes, so metrics compare across subject areas.
- Slowly changing dimension — a technique for storing how descriptive attributes change over time, either by overwriting or by keeping versioned rows.
- Data vault — a modelling method that separates business keys, relationships and descriptive history into hubs, links and satellites.
- Data contract — an agreement between a data producer and its consumers that defines schema, meaning, quality expectations and change rules.

Data Warehouse and Lakehouse Engineering Day by Day

Day 1: The Analytical Storage Landscape

- Warehouse, Lake and Lakehouse Patterns — Choosing a Target Storage Paradigm
- Separation of Storage and Compute — Scaling Query Capacity Independently
- Open Table Formats — Engine Interoperability and Vendor Independence
- Medallion Layering of Bronze, Silver and Gold — Trust Boundaries Between Layers
- Workload Profiling of Queries and Users — Sizing the Storage Estate

Day 2: Dimensional Modelling Foundations

- Bus Matrix of Processes and Dimensions — Prioritising Subject Areas
- Fact Table Grain Declaration — Consistent Aggregation Across Reports
- Transaction, Periodic and Accumulating Snapshot Facts — Matching Process Behaviour
- Conformed Dimensions and Surrogate Keys — Cross-Domain Metric Comparison
- Role-Playing and Junk Dimensions — Simplifying Analyst Query Paths

Day 3: History and Change in Dimensions

- Slowly Changing Dimension Types One and Two — Restating or Preserving History
- Mini-Dimensions for Volatile Attributes — Controlling Dimension Table Growth
- Bridge Tables for Many-to-Many Relationships — Correct Allocation of Measures
- Late-Arriving Facts and Dimensions — Integrity of Historical Figures
- Snapshot Tables and Time Travel Queries — Audit and Point-in-Time Reporting

Day 4: Data Vault Modelling

- Hubs and Business Keys — Stable Integration Across Source Systems
- Links and Relationship Grain — Capturing Associations Without Rework
- Satellites and Load Metadata — Auditable Descriptive History
- Point-in-Time and Bridge Helper Tables — Query Performance Over the Vault
- Business Vault Derivations — Separating Raw Records from Applied Rules

Day 5: Physical Table Design

- Partitioning Strategies by Date and Key — Pruning Scanned Files
- Clustering, Z-Ordering and Sort Keys — Selective Query Speed
- File Sizing and Compaction — Avoiding the Small Files Problem
- Columnar File Formats and Compression — Storage Footprint Versus Read Cost
- Materialised Views and Aggregate Tables — Serving Frequent Questions Cheaply

Day 6: Lakehouse Table Mechanics

- Transaction Logs and Snapshot Isolation — Concurrent Readers and Writers
- Schema Evolution and Enforcement Rules — Safe Column Changes
- Merge, Upsert and Delete Semantics — Record-Level Corrections and Erasure Requests
- Snapshot Expiry and Vacuum Policies — Retention Versus Recoverability
- Catalogues and Table Metadata Services — Discoverability Across Engines

Day 7: Data Contracts and Semantic Agreements

- Contract Specifications for Schema and Meaning — Producer and Consumer Responsibilities
- Freshness and Completeness Expectations — Defining Acceptable Table Staleness
- Breaking Change Classification and Versioning — Negotiating Schema Changes
- Semantic Layer and Metric Definitions — One Answer per Business Question
- Domain Ownership of Gold Tables — Accountability for Published Data Products

Day 8: Access, Security and Platform Health

- Row-Level and Column-Level Access Policies — Protecting Sensitive Attributes
- Tagging and Classification of Tables — Applying Retention and Masking
- Table Health Indicators and Freshness Monitors — Early Signals of Stale Data
- Column Lineage Across Layers — Impact of Upstream Schema Changes
- Query History and Slow Query Diagnosis — Targeting Redesign Effort

Day 9: Storage Cost Engineering

- Compute Consumption Attribution by Domain — Chargeback and Showback Choices
- Storage Tiering and Lifecycle Rules — Hot, Warm and Archive Placement
- Warehouse Sizing and Auto-Suspend Policies — Idle Spend Reduction
- Cost per Query and per Table Metrics — Prioritising Optimisation Work
- Migration Paths from Legacy Warehouses — Sequencing a Platform Move

Day 10: Practising the Storage Architecture Decision Record

- Paradigm Selection for a Retail Sales Domain — Applying Pattern Trade-Offs
- Star Schema and Vault Design for a Utilities Meter Domain — Applying Both Methods
- Contract and Access Policy Drafting — Applying Agreements to Gold Tables
- Layout and Cost Projection for the Target Design — Applying Cost Engineering
- Storage Architecture Decision Record — Completing and Defending the Target Design

Applying Data Warehouse and Lakehouse Engineering After the Course

- Lead a grain and conformance review of one subject area using the bus matrix, and publish the agreed conformed dimensions.
- Run a table layout audit on the ten most expensive tables, recommending partitioning, clustering and compaction changes with projected cost effect.

Data Warehouse and Lakehouse Engineering Questions FAQ

What is data warehouse and lakehouse engineering and why does it matter now?

It is the discipline of designing how analytical data is modelled, laid out and governed in storage. It matters now because open table formats have merged lake and warehouse capabilities, while compute bills and audit expectations make every storage choice visible to leadership.

Which roles gain most from data warehouse and lakehouse engineering?

Data engineers, modellers, architects and platform owners gain most. They decide table structure, history handling, access and storage spend.

How does data warehouse and lakehouse engineering differ from a data integration course?

It focuses on what data looks like at rest rather than how it moves. You concentrate on dimensional and vault models, table mechanics, contracts and cost, while integration courses concentrate on extraction, scheduling and movement between systems.

What do you bring back from data warehouse and lakehouse engineering?

You bring back a written storage architecture decision for a real domain. It records the chosen paradigm, the dimensional and vault models, table layout, contracts, access policies and projected cost.

Data Warehouse and Lakehouse Engineering: The Way Forward

The Storage Architecture Decision Record gives architects and platform owners a defended target for one analytical domain. Its first use is the choice between consolidating on a lakehouse or keeping a separate warehouse tier. Revisit its cost projections after the first quarter of real query traffic, when actual consumption replaces estimates.