**Ecommerce Data Vault & Delta Lake Project Case Study**

**1. Problem Statement**

ShopFast, a growing ecommerce company, needs a modern, scalable, and compliant data platform that enables real-time analytics on customer behaviour, product sales, and inventory. The platform must support:

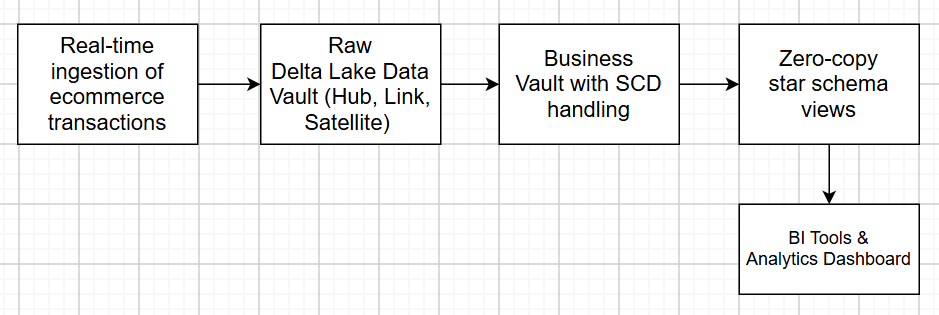
* A flexible data model that can evolve with business needs
* Historical data tracking with time travel and slowly changing dimensions
* GDPR-compliant data handling and deletion
* High-performance query capability with Delta Lake optimizations
* Automated, auditable data vault pipelines with surrogate key management
* Business-friendly star schemas for reporting and BI

**2. Skill Tower Developed by the Project**

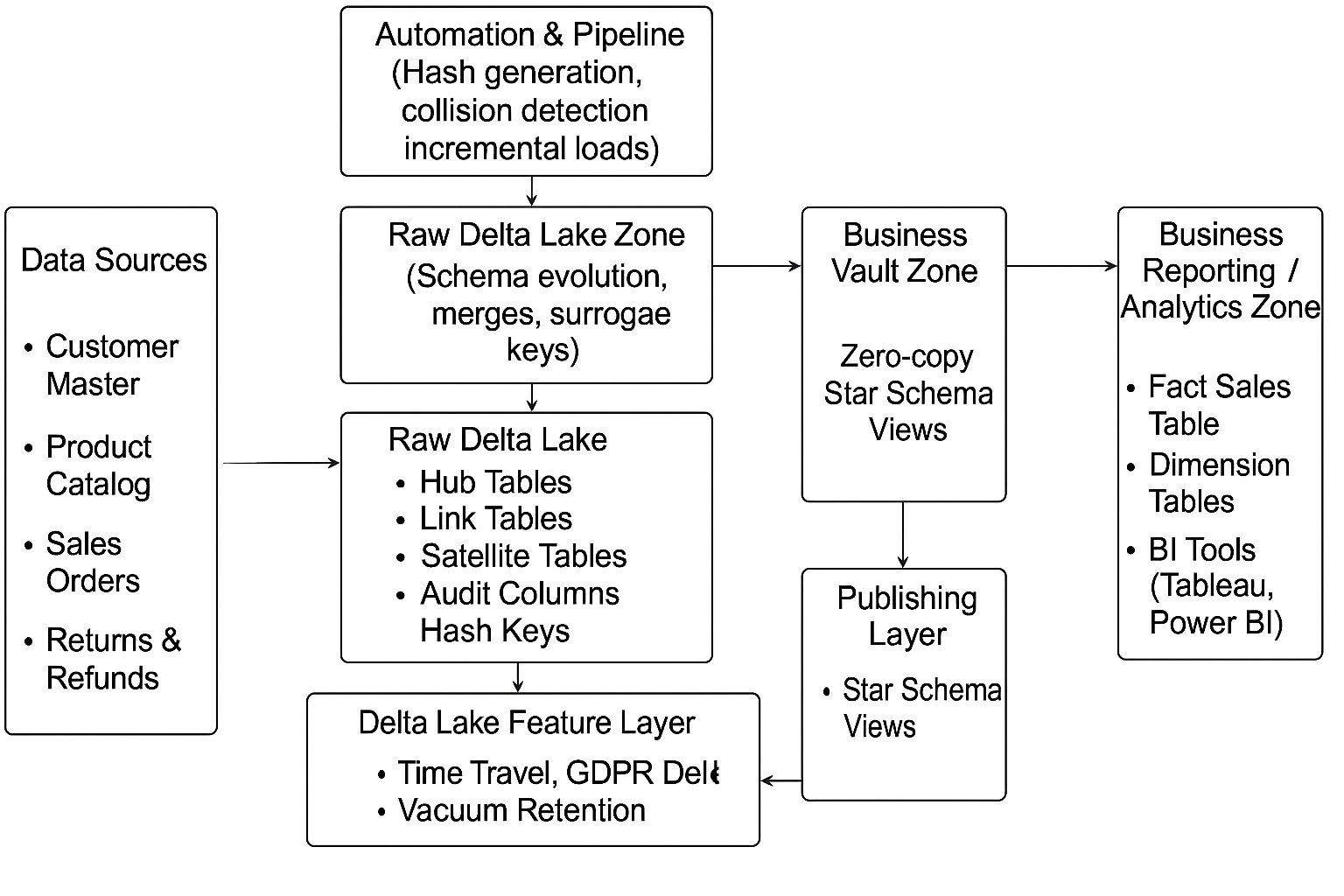
| **Skill Area** | **Description** |
| --- | --- |
| Data Modeling | Designing Hub/Link/Satellite, Star Schemas, ERDs, Grain |
| Data Vault Methodology | Hash keys, collision handling, PIT & Bridge tables |
| Delta Lake Features | Schema evolution, generated columns, OPTIMIZE ZORDER, VACUUM |
| Data Engineering Automation | Pipeline scripting, surrogate key generation, macros |
| Data Governance & Compliance | GDPR deletion patterns, audit columns, time-travel |
| SQL & Spark Programming | Complex SQL transformations, data merges, Delta table ops |
| Version Control & Collaboration | Git versioning, peer review, documentation |

**3. Use Case / Architecture Diagram**

**Use Case:** Real-time ingestion of ecommerce transactions → Raw Delta Lake Data Vault (Hub, Link, Satellite) → Business Vault with SCD handling → Zero-copy star schema views → BI Tools & Analytics Dashboard



**Architecture Diagram:**



**4. User Stories**

* **As a data engineer,** I want to design a Data Vault schema to store raw ecommerce data with auditability and hashing, so that I can ensure data integrity and lineage.
* **As a data engineer,** I want to implement Delta Lake schema evolution to accommodate new business attributes without downtime.
* **As a data analyst,** I want to query business-friendly star schema views that reflect accurate and historical sales data.
* **As a compliance officer,** I want to implement GDPR-compliant data deletion for customers on request.
* **As a data engineer,** I want to automate surrogate key generation and data loading pipelines for consistency and efficiency.
* **As a data engineer,** I want to optimize data layout with ZORDER and VACUUM to improve query performance and control storage costs.

**5. Expected Deliverables**

* ERDs for raw Data Vault (Hub/Link/Satellite) and star schema with naming standards (versioned in Git)
* Delta Lake raw vault tables with evolving schema, generated columns, and audit columns
* Scripts demonstrating Delta Lake features: OPTIMIZE ZORDER, VACUUM retention, time-travel restore
* Implementation of GDPR delete pattern with PII masking/deletion scripts
* SCD Type 0-6 implementation scripts, surrogate key generation automation
* PIT and Bridge tables scripts for performance optimization
* Data Vault automation macros/scripts for hash keys, loading, and collision detection
* Published zero-copy star schema views for BI consumption
* Documentation covering grain declaration, SCD checklist, and peer-review feedback
* Git repository containing all scripts, models, documentation, and version history

**6. Milestones and Duration**

| **Milestone** | **Duration** | **Description** |
| --- | --- | --- |
| Requirements Gathering & ER Modeling | 4 hours | Interview, define grain, ERDs, version in Git |
| Build Raw Data Vault & Schema Evolution | 4 hours | Create Delta tables, hash keys, evolve schema |
| Optimize & Retain Data, GDPR Demo | 2 hours | OPTIMIZE, VACUUM, time-travel restore, GDPR deletion |
| Implement SCD & Surrogate Keys | 4 hours | SCD types, surrogate key automation, PIT/Bridge tables |
| Automate Pipelines & Publish Star Schema | 3 hours | Automation scripts, zero-copy star schema publishing |
| Documentation & Peer Review | 1 hour | Final documentation, checklist, and model/code reviews |

**7. Implementation Notes**

* Use Apache Spark with Delta Lake for data storage and transformation.
* Use hash functions (e.g., SHA-256) for key generation and collision detection.
* Apply Delta Lake schema evolution (ALTER TABLE ... ADD COLUMNS) carefully during satellite updates.
* For GDPR delete, perform Delta Lake update/delete with filtering on PII columns.
* Use ZORDER to cluster data on high-cardinality columns used in queries (e.g., order\_date, product\_id).
* VACUUM retention should balance between storage cost and ability to time-travel rollback.
* Surrogate key generation should be atomic and idempotent; implement as a reusable macro or script.
* Peer review models and code with version control (Git) to ensure quality and collaboration.

**8. Evaluation Rubrics**

| **Criteria** | **Excellent (4)** | **Good (3)** | **Fair (2)** | **Poor (1)** |
| --- | --- | --- | --- | --- |
| ER Modeling | Complete, accurate ERDs, versioned | Mostly accurate ERDs with minor gaps | Partial ERDs or missing version control | Missing or inaccurate ERDs |
| Data Vault & Delta Implementation | Correct hubs/links/satellites, hash keys, schema evolution fully handled | Minor issues in DV or schema evolution | Partial implementation, inconsistent hash keys | Missing key components |
| Delta Features | Full use of OPTIMIZE, VACUUM, time travel, metadata tables | Partial implementation with minor gaps | Limited use, incorrect implementation | No Delta features used |
| SCD & Surrogate Keys | SCD 0-6 implemented, surrogate keys automated | Basic SCD and surrogate keys | Partial or incorrect SCD | No SCD or surrogate key implementation |
| Automation & Pipelines | Well-structured, reusable scripts/macros | Some automation scripts | Minimal automation | No automation |
| GDPR & Data Management | Correct delete patterns, audit columns | Partial GDPR compliance | Minimal or incorrect GDPR handling | No GDPR or compliance handling |
| Documentation & Version Control | Complete, clear docs, peer review, Git versioning | Mostly complete docs, basic version control | Incomplete docs or limited versioning | No docs or version control |