Course Summary Report: DP-750 Implement Data Engineering Using Azure Databricks

1. Executive Course Overview

The DP-750 course provides a rigorous, end-to-end framework for implementing production-grade data engineering solutions on the Azure Databricks platform. Designed for the modern Lakehouse architecture, the curriculum is unified under the **Unity Catalog** framework, ensuring that every stage of the data lifecycle—from raw ingestion to analytical serving—is governed, secure, and observable.Throughout the course, students navigate real-world industry scenarios ranging from telecommunications to healthcare, focusing on five primary architectural pillars:

- **Data Ingestion:** Leveraging batch, incremental, and continuous streaming patterns.
- **Medallion Architecture:** Implementing Bronze, Silver, and Gold layers to ensure multi-stage data refinement.
- **Governance:** Centralizing metadata, tracking deep lineage, and enforcing compliance.
- **Security:** Utilizing fine-grained access controls, row/column filters, and secret management.
- **Pipeline Orchestration:** Automating complex, interdependent workflows with robust error handling and monitoring.

2. Module 1: Workspace Navigation and Data Ingestion Basics

Setting the foundation within the **CityMoves Transit** scenario, this module introduces the Databricks Workspace UI as the primary environment for a data engineer.

- **Sidebar Navigation:** Core navigation centers on **Workspace** (asset management), **Recents** , **Catalog** (Unity Catalog Explorer), **Compute** , and **Jobs & Pipelines** .
- **Genie Code:** The curriculum emphasizes the use of **Genie Code** , an AI-powered assistant, for real-time code generation, troubleshooting, and logic explanation within notebooks.
- **Data Ingestion UI:** Foundational data movement involves using the "Add or upload data" interface to move local datasets (e.g., transit route CSVs) into **Unity Catalog volumes** , which provide a managed storage layer for unstructured or semi-structured files.

3. Module 2: Compute Configuration, Clusters, and Libraries

Using the **HealthBridge Analytics** scenario, this module explores the trade-offs in compute selection for healthcare workloads involving millions of patient records.

- **All-Purpose Clusters:** Configuration focuses on stability and performance, specifically utilizing **Databricks Runtime LTS** versions and **Photon Acceleration** for vectorized query execution. For heavy join operations common in healthcare, memory-optimized worker types (e.g., **Standard_DS12_v2**) are recommended.
- **Library Management:**

- **Cluster-scoped:** Installed via the UI for persistence across restarts and availability to all users.
- **Notebook-scoped:** Utilizes %pip for session-specific dependencies, ensuring isolated development environments.
- **Synthetic Data Generation:** The **"faker"** library is introduced as a best practice for generating realistic, PII-compliant test data.
- **Governance & Cost Control:** Implementation of **autoscaling** and **auto-termination** (e.g., a 15-minute idle timeout) is required to align with strict organizational budget governance.

4. Module 3: Unity Catalog Organization

This module implements the full namespace hierarchy for **Lakeside University** : **Catalog > Schema > Table/Volume** .

- **Medallion Implementation:** Students build out the university's academic data platform, moving from raw Bronze ingestion to Silver (cleaned tables with primary/foreign key constraints) and finally Gold (analytical aggregates).
- **Managed Objects:** The architecture prioritizes **Managed Tables** , as they allow Unity Catalog to handle both metadata and underlying file maintenance automatically.
- **Advanced Objects:** Includes the creation of standard views for logic encapsulation, materialized views for performance, and managed volumes for file storage.
- **AI/BI Genie Space:** An introduction to natural language querying, allowing non-technical stakeholders to query university enrollment statistics using plain English.

5. Module 4: Security and Access Control

Focusing on **NorthMart Retail** , this module details how to secure a centralized data platform across multiple geographic regions.

- **Permission Models:** Managing access at the schema level by granting specific privileges to identity-based groups, such as retail-analysts.
- **Fine-Grained Security:**
- **Row Filters:** Implementing functions to restrict record visibility based on the user's assigned region (e.g., North vs. West).
- **Column Masks:** Protecting PII by masking sensitive fields like email addresses or loyalty API keys.
- **Secret Management:** Integration with **Azure Key Vault** to create **Secret Scopes** . This allows engineers to retrieve credentials securely in notebooks using dbutils.secrets.get(), preventing the exposure of sensitive keys in source code.

6. Module 5: Governance and Lineage

The **AutoSphere AG** scenario highlights the importance of traceability in a connected vehicle platform.

- **Data Discovery:** Utilizing descriptive comments and tags within the Catalog Explorer to facilitate asset discovery across the organization.

- **GDPR & Retention:** Implementation of Delta Lake retention policies and the **VACUUM** operation to remove expired data files, ensuring compliance with data minimization requirements.
- **Lineage Tracking:** Visualizing table-level and column-level lineage to trace telemetry data from its source to downstream dashboards.
- **System Tables:** Querying built-in system tables programmatically to audit data access patterns and investigate lineage for compliance reporting.

7. Module 6: Data Modeling and Advanced Delta Lake Features

In the **Northbank Financial** scenario, the curriculum addresses regulatory requirements for historical accuracy and auditability.

- **Table Architecture:** A comparison of **Managed vs. External tables** , with a recommendation for Managed tables at Northbank to benefit from automatic maintenance and simpler governance.
- **Performance Optimization:** Implementation of **Liquid Clustering** on large transaction fact tables to replace traditional partitioning and accelerate date-range filtering.
- **Historical Tracking:**
- **SCD Type 2:** Maintaining a history of changing customer profiles using the **MERGE** statement.
- **Change Data Feed (CDF):** Enabling a queryable log of all modifications for audit trails via table_changes().
- **Time Travel:** Utilizing point-in-time recovery to restore or inspect previous table versions.

8. Module 7: Data Ingestion Techniques

Working with **Solaris Energy** , this module covers technical patterns for ingesting wind and solar telemetry from disparate global sites.

- **Ingestion Patterns:**
- **PySpark DataFrames:** Used for complex batch loading logic.
- **SQL "COPY INTO":** Employed for idempotent, incremental file loading with built-in deduplication.
- **CTAS:** Creating summary tables for rapid reporting.
- **Auto Loader:** The primary tool for continuous file detection in cloud storage, providing schema evolution and efficient processing.
- **Lakeflow Spark:** Introduction to declarative pipelines (formerly Delta Live Tables) as the gold standard for production ETL.

9. Module 8: Data Cleansing and Transformation

The **Pristine Properties** scenario involves refining messy real estate listings into high-quality assets.

- **Data Profiling:** Utilizing the **"Quality" tab** in Catalog Explorer to generate snapshot profiles, identifying null distributions and unexpected values in property prices.

- **Transformation Logic:** Applying type-casting, null handling, and record deduplication.
- **Reshaping Data:** Using Spark **Pivot** and **Unpivot** operations to transform wide-format satellite data into long-format statistics for market trend analysis.

10. Module 9: Data Quality with Lakeflow Spark

Focusing on **ClearCover Insurance** , this module implements Lakeflow Spark Declarative Pipelines to catch malformed insurance claims before they reach actuarial models.

- **Expectations Hierarchy:**| Expectation Level | Action Taken | Use Case || ------ | ------ | ------ || **expect** | Warn only | Logs violations for non-critical fields; keeps the record. || **expect_or_drop** | Cleanse data | Drops records that violate critical constraints (e.g., missing IDs). || **expect_or_fail** | Halt pipeline | Stops the entire job if a critical quality threshold is not met. |
- **Schema Drift:** Configuring Auto Loader with schemaEvolutionMode rescue to capture unexpected fields (like a new broker reference) into a _rescued_data column without breaking the pipeline.

11. Module 10: Pipeline Design and Orchestration

This module orchestrates an end-to-end Medallion architecture for **GlobStay** hospitality bookings.

- **Modular Design:** Using **Lakeflow Jobs** to stitch together sequential tasks (Bronze > Silver > Gold).
- **Parameterization:** Implementing **dbutils.widgets** for reusable notebook logic and **taskValues** to pass dynamic values between job stages.
- **Error Handling:** Using try/except blocks combined with dbutils.notebook.exit() to signal specific status codes to the job orchestrator.
- **Conditional Flows:** Implementing **If/else condition** tasks to route the pipeline (e.g., triggering an alert if a data quality threshold is exceeded).

12. Module 11: Lakeflow Jobs Automation

The **TelConnect** scenario demonstrates operationalizing a high-volume Call Detail Record (CDR) pipeline.

- **Orchestration Triggers:** A production best practice is established by combining **File Arrival triggers** (for event-based processing as CDRs land in a volume) with **Scheduled triggers** (using **Cron expressions** like 0 0 2 * * ?) to provide a nightly catch-all refresh.
- **Operational Reliability:**
- **Muted Retries:** Configuring notifications to only alert the team if the final retry attempt fails, reducing alert noise.
- **Automatic Retries:** Setting a policy of 2 retries with 60-second intervals to handle transient cloud storage errors.
- **Timeouts:** Implementing a 30-minute timeout threshold to prevent stalled tasks from hanging indefinitely.

13. Module 12: Software Development Lifecycle (SDLC)

Professional engineering standards are applied to an order processing pipeline.

- **Testing:** Utilizing **pytest** and fixtures to implement unit tests for transformation functions and integration tests for full pipeline validation.
- **Infrastructure-as-Code:** Packaging the solution using **Databricks Asset Bundles (DABs)** , with all resources defined in a **databricks.yml** configuration file.
- **Databricks CLI:** Using the CLI to **validate** , **preview** , and **deploy** the bundle to specific dev or prod targets, ensuring repeatable and auditable deployments.

14. Module 13: Performance Monitoring and Optimization

The final module focuses on diagnosing the "hallmark" problems of **Data Skew** and **Excessive Shuffle** .

- **Troubleshooting Environment:** In this module, students must **disable Photon** (the vectorized query engine). Photon is often too efficient, masking the performance problems that the Spark UI is designed to reveal for educational purposes.
- **Spark UI Diagnostics:** Analyzing the Jobs and Stages tabs to identify skewed task distributions (where a few tasks take significantly longer than the median) and excessive shuffle (where network I/O far exceeds input size).
- **Optimization Techniques:** Applying targeted fixes including **Broadcast joins** for small table lookups, **Adaptive Query Execution (AQE)** , and re-enabling **Photon** for production optimization.

15. Conclusion and Key Takeaways

The DP-750 curriculum equips architects with the skills to build robust, scalable, and governed data platforms. The three most critical takeaways are:

1. **Unity Catalog is the Governance Anchor:** All modern Databricks features, from security filters to lineage and asset bundles, depend on a well-structured Unity Catalog implementation.
2. **The Medallion Architecture ensures Quality:** Moving from Bronze to Gold is not just about storage, but about a systematic increase in data reliability and analytical readiness.
3. **Production Readiness requires Automation and SDLC:** Reliable pipelines depend on integrated testing, robust Lakeflow Jobs orchestration with multiple trigger types, and Infrastructure-as-Code deployment via Asset Bundles.