DATA ENGINEERING | ADVANCED | 2 DAYS

Databricks SQL and Migration from T-SQL

Migrate T-SQL workloads to Databricks SQL — map syntax differences, rewrite scripts for the lakehouse, and tune queries with warehouses, Photon, and Delta performance patterns.

Why Databricks SQL?

SQL on the lakehouse

Work natively in Databricks SQL warehouses, the SQL editor, and query history.

Performance with Photon

Read query profiles and write SQL that leverages Delta file skipping and warehouse compute.

Confident T-SQL migration

Translate temporary tables, functions, types, and collation behaviour without guesswork.

Production-ready tuning

Apply partitioning, Z-Ordering, OPTIMIZE, liquid clustering, and caching patterns.

What you'll be able to do

- ✓ Write and execute SQL queries natively in the Databricks SQL environment
- ✓ Map key syntax differences between T-SQL and Databricks SQL with practical examples
- ✓ Handle migration-specific challenges: temporary tables, case sensitivity, string collation, and date functions
- ✓ Rewrite existing T-SQL scripts to run correctly and efficiently in Databricks SQL
- ✓ Connect clients and BI tools to Databricks SQL using HTTP path, tokens, and JDBC/ODBC
- ✓ Read and interpret query plans and the Databricks SQL query profile
- ✓ Apply SQL optimisation techniques: partitioning, Z-Ordering, OPTIMIZE, liquid clustering, and caching
- ✓ Apply best practices for writing performant SQL in a lakehouse environment

Who this is for

- Data Engineers
- Data Analysts
- Analytics Engineers
- BI Professionals
- Cloud Engineers
- Data Platform Teams

Curriculum

01 Databricks SQL Workspace and Warehouses

- SQL editor, SQL warehouses, query history, and dashboards in the Databricks SQL workspace
- Warehouse types: Classic versus Pro versus Serverless — differences and when to use each
- How warehouse sizing and autoscaling affect interactive and scheduled SQL workloads
- Navigating query history to review past runs and share SQL with teammates
- Organising SQL assets for migration projects and recurring analytics

02 Connecting Clients to Databricks SQL

- Connection essentials: server hostname, HTTP path, and authentication options
- Personal access tokens versus other auth patterns for SQL clients
- JDBC and ODBC connectivity for desktop tools and applications
- Partner Connect and common BI tool patterns for Databricks SQL
- Checklist for secure, least-privilege access from migration source systems

03 Mapping T-SQL Syntax to Databricks SQL

- Side-by-side mental model: what stays familiar and what must change
- Row limiting: TOP versus LIMIT and ORDER BY pairing rules
- Case sensitivity for identifiers versus string comparisons
- Common migration gotchas that break scripts on first run
- Building a personal T-SQL to Databricks SQL cheat sheet during the course

04 Temporary Objects, Identity Columns, and Data Types

- Temporary tables: CREATE #temp versus CREATE OR REPLACE TEMP VIEW and session scoping
- IDENTITY columns versus GENERATED ALWAYS AS IDENTITY
- Data type mapping: NVARCHAR versus STRING, DATETIME2 versus TIMESTAMP, MONEY versus DECIMAL
- Choosing safe casts when rewriting typed T-SQL scripts
- Validating migrated DDL before rewriting dependent queries

05 Strings, Dates, Collation, and NULL Semantics

- Date and time functions: GETDATE() versus CURRENT_TIMESTAMP() and DATEADD equivalents
- String functions: LEN() versus LENGTH(), CHARINDEX() versus LOCATE(), ISNULL() versus IFNULL()/COALESCE()
- UTF8_LCASE collation for case-insensitive matching — syntax, use cases, and migration implications
- NULL comparison behaviour differences that change filter results
- Practical rewrite patterns for reporting queries that rely on T-SQL null and string defaults

06 Migrating CTEs, Subqueries, and Window Functions

- CTE and subquery compatibility notes when moving from T-SQL
- Window function syntax that ports cleanly versus patterns that need adjustment
- Preserving business logic while simplifying nested SQL for Databricks
- Testing migrated analytical queries for row-level parity
- **Hands-on:** Migrate a set of T-SQL scripts to Databricks SQL with guided corrections and testing

07 Query Execution, Photon, and the Query Profile

- How Databricks SQL executes queries: warehouse compute, Photon, and query compilation
- Reading the query profile: stages, tasks, spill, and bottleneck identification
- Linking profile symptoms to SQL and table-layout fixes
- Using query history metrics to prioritise optimisation work
- Baseline a slow query before and after each tuning change

08 Writing Efficient SQL on Delta Lake

- Avoid SELECT *: project only required columns to reduce scan width
- Predicate pushdown and filters that enable Delta file skipping
- Partition pruning: writing predicates that reduce data scanned
- Prefer built-in functions over UDFs for Photon compatibility and speed
- Use MERGE INTO carefully; know when a full overwrite is more efficient
- Avoid unnecessary DISTINCT and ORDER BY on large sets without LIMIT

09 Table Layout: Partitioning, Z-Order, OPTIMIZE, and VACUUM

- Partitioning strategy: column choice, cardinality, and anti-patterns
- Z-Ordering: choosing columns and running ZORDER BY to co-locate related data
- OPTIMIZE for the small-file problem — when to run and how to schedule it
- VACUUM: safe retention, time travel trade-offs, and scheduling
- Liquid clustering versus partitioning and Z-Ordering, plus a practical migration path
- Result cache and disk cache behaviour in Databricks SQL

10 Capstone: Migrate and Optimise a T-SQL Workload

- Rewrite a representative T-SQL script set to run correctly in Databricks SQL
- Validate functional parity for filters, joins, and aggregations
- Profile slow queries and apply layout or SQL fixes (Z-Order, OPTIMIZE, predicates)
- Document remaining risks: collation, types, and warehouse sizing
- **Hands-on:** Deliver migrated, profiled SQL ready for a Databricks lakehouse environment