**🗓️ WEEK 1 — *Data Engineering SQL Foundations***

**Goal:** Shift from “writing queries” to “building transformation pipelines.”  
You’ll learn how to structure multi-step SQL processes cleanly and efficiently.

**Day 1 — Think in Sets, Not Steps**

**Theory (15 min)**

* SQL is *declarative*: you describe *what* you want, not *how* to get it.
* Contrast procedural logic (loops) vs set logic.
* SQL Server query execution order:  
  FROM → WHERE → GROUP BY → HAVING → SELECT → ORDER BY.

**Practice (20 min)**

* Write one query that aggregates total sales by customer.
* Then split it into logical steps using **CTEs**:
* WITH base AS (
* SELECT CustomerID, SUM(Amount) AS total\_sales
* FROM Sales
* GROUP BY CustomerID
* )
* SELECT \* FROM base WHERE total\_sales > 5000;

**Mini-Task (5 min)**  
Compare readability between single-query vs step-wise CTE.

**Day 2 — Master Common Table Expressions (CTEs)**

**Theory (10 min)**

* When to use CTEs vs subqueries vs temp tables.
* SQL Server materialization concept.

**Practice (25 min)**  
Build a 3-step pipeline:

1. Clean invalid data (WHERE conditions).
2. Aggregate sales by region.
3. Filter top regions.  
   Use chained CTEs.

**Mini-Task (5 min)**  
Check if query plans show re-computation or materialization.

**Day 3 — Temp Tables & Table Variables**

**Theory (15 min)**

* #TempTables vs @TableVariables — scope, performance, indexing.
* When to persist intermediate data for big transformations.

**Practice (20 min)**  
Run the same logic using:

1. A CTE
2. A Temp Table  
   Compare execution times via “Actual Execution Plan.”

**Mini-Task (5 min)**  
Note which one’s faster and why.

**Day 4 — Transactions, ACID, and Error Handling**

**Theory (15 min)**

* ACID in context of ETL loads.
* Transaction scope, rollback, and try/catch in SQL Server.

**Practice (20 min)**  
Simulate an ETL insert:

1. Start TRAN.
2. Insert rows.
3. Intentionally fail one row.
4. ROLLBACK and check results.

**Mini-Task (5 min)**  
Write your own TRY...CATCH block around insert/update.

**Day 5 — Multi-Step Transformation Project**

**Goal:** Combine what you learned.

**Task (40 min)**  
Create a pipeline:

1. **Stage data** in a temp table.
2. **Clean**: remove duplicates, nulls.
3. **Join** with a lookup table.
4. **Aggregate** metrics by category.
5. **Output** to a reporting table.

**Deliverable:**  
A single SQL script using CTEs + transactions that runs end-to-end.

**🗓️ WEEK 2 — *Advanced Joins & Conditional Logic***

**Goal:** Master join mechanisms and learn how to make SQL “think conditionally.”

**Day 1 — Deep Dive: JOIN Mechanics**

**Theory (15 min)**

* INNER vs LEFT vs RIGHT vs FULL
* Nested Loops, Hash Joins, Merge Joins — how SQL Server picks one.

**Practice (20 min)**

* Use SET SHOWPLAN\_ALL ON and examine plans for different join types.
* Compare performance joining large vs small tables.

**Mini-Task (5 min)**  
Explain why order of joins can matter.

**Day 2 — Conditional Joins**

**Theory (10 min)**

* Use conditional logic in join predicates.
* Date-range joins and active-record joins.

**Practice (25 min)**  
Join Orders to Promotions where:

o.OrderDate BETWEEN p.StartDate AND p.EndDate

Then add conditionally active promotions only.

**Mini-Task (5 min)**  
Write query showing only *currently active* promotions.

**Day 3 — CASE Expressions for Business Logic**

**Theory (10 min)**

* Use CASE in SELECT and WHERE.
* Dynamic grouping and flagging.

**Practice (25 min)**  
Build a report that:

* Categorizes orders as “Low/Med/High Value.”
* Flags customers with declining sales.

**Mini-Task (5 min)**  
Add a derived column: CustomerStatus.

**Day 4 — Conditional Aggregations**

**Theory (10 min)**

* SUM(CASE WHEN … THEN … END) for pivot-style metrics.
* Difference between HAVING and WHERE.

**Practice (25 min)**  
Create a table of:

* Total Orders,
* Cancelled Orders,
* Return Rate = SUM(CASE …)/COUNT(\*).

**Mini-Task (5 min)**  
Turn that into a view.

**Day 5 — Advanced Join Patterns & Mini Project**

**Theory (10 min)**

* Anti-join: NOT EXISTS
* Semi-join: EXISTS
* Self-joins for hierarchy or “previous record” lookups.

**Practice (25 min)**

* Find customers who purchased but never returned (NOT EXISTS).
* Find most recent order per customer (self-join).

**Mini-Project (5 min)**  
Build a *Business Metrics View* with:

* Total sales
* Repeat customers
* Active customers (last 30 days)
* New customers (current month)

✅ **End of Week 2 Outcome**  
You’ll be able to:

* Write efficient join logic (and know why it’s efficient).
* Translate messy business rules into conditional SQL.
* Build analytical views ready for dashboards or downstream ETL.