Lecture Notes – RecursiveCTEs

Section 1: Lecture Summary

Recursive **CTEs** (Common Table Expressions) use the **WITH RECURSIVE** clause to generate sequences like numbers from 1 to 10 or days from 1 to 30. A recursive CTE consists of a starting SELECT (anchor) combined with a recursive SELECT using **UNION ALL**, referencing the CTE itself in the FROM clause, with a **WHERE** condition for termination.

Section 2: Key Concepts and Explanations

CTEs are named SQL queries defined with **WITH** and used in the main query's FROM clause. Recursive CTEs require:

- **Anchor member**: Initial SELECT (e.g., **SELECT 1 AS N**) for the starting point.

- **Recursive member**: SELECT referencing the CTE (e.g., **SELECT N + 1 FROM numbers**), combined via **UNION ALL**.

- **Termination condition**: WHERE clause (e.g., **WHERE N < 10**) to stop recursion.

Union parts must have compatible columns (same number and data types). Recursive CTEs mimic recursive functions, self-referencing until termination. Useful for generating sequences or hierarchical data.

Section 3: Example Code and Use Cases

```
WITH RECURSIVE numbers (N) AS (
  SELECT 1
  UNION ALL
  SELECT N + 1 FROM numbers WHERE N < 10
)
SELECT N FROM numbers;
```

Generates numbers 1 to 10.

```
WITH RECURSIVE days (N) AS (
  SELECT 1
  UNION ALL
  SELECT N + 1 FROM days WHERE N < 30
```

```
)
SELECT N FROM days;
```

Generates numbers 1 to 30 for days in a month.

Section 4: Key Takeaways

- Start with anchor SELECT, recurse with self-reference via **UNION ALL**, terminate with WHERE.

- Columns must match in union for compatibility.

- Practice generating sequences; apply to hierarchies next.