Advanced SQL – Proposed Curriculum

Topics:

Day 1, 2, 3, 4

1. Relationship and Cardinality
    a. One - to – One
    b. One - to – Many
    c. Many - to – Many

This topic may get covered while explaining sample database(s).

2. Joins
    a. Inner Join
    b. Equi Join
    c. Natural Join
    d. Non-Equi Join
    e. Self-Join
    f. Outer Join
        i. Left Outer Join
        ii. Right Outer Join
        iii. Full Outer Join
    g. Cross Join

Following scenarios to be discussed:

1. Pick up just 1 record from rhs (say on datetimestamp or rowno) when multiple records exists
2. Pickup all records from rhs table
3. Create an intermediate table having rolled up data
    a. ✅ **2019**:
        i. Introduce APPLY (CROSS APPLY, OUTER APPLY)
        ii. Use of ROW_NUMBER() inside JOINs for de-duplication
    b. ✅ **2022**:
        i. Combine JOINs with DATE_BUCKET() for event tracking
        ii. Discuss join performance improvements via IQP features

3. Built In Functions
    a. Scalar Functions
        i. Numeric Functions

ii. String Functions

1. ✅ **2019**:
   a. **STRING_AGG()**
   b. **TRIM(), CONCAT_WS(), TRANSLATE(), STRING_SPLIT() (improved)**
2. • ✅ **2022:**
   a. **DATE_BUCKET() (new for time-series grouping)**
   b. **Enhanced STRING_SPLIT() with ordinal column**
   c. **IS DISTINCT FROM for NULL-safe comparisons**

iii. Conversion Functions

iv. Date Functions

v. Aggregate Functions

vi. Convenient Aggregate Functions

vii. Statistical Aggregate Functions

viii. Super Aggregates

b. OVER and PARTITION BY Clause

   a. Ranking Functions

c. Top n Clause

   i. **With TIES, Order-based filters**

Important functions/elements to be covered Lead, Lag, all rank functions, when to use which, firstvalue, rowno.

4. Set Operators
   a. Union
   b. Intersect
   c. Except

- ✅ **Include performance considerations and NULL handling behaviour in 2019/2022**

Day 4 (partial), 5, 6

5. SUBQUERIES
   a. Single Row Sub Queries
   b. Multi Row Sub Queries
   c. Built-in function
   d. Nested Sub Queries

e. Correlated Sub Queries

f. Derived tables

g. Recursive queries

h. Dynamic Queries and Pivots

i. Common Table Expression

Day 7, 8, 9

6. Views

   a. Purpose of Views

   b. Creating, Altering and Dropping Indexes

   c. Simple and Complex Views

   d. Encryption and Schema Binding Options in creating views

- ✅ 2019**: Indexed Views limitations with STRING_AGG(), JSON**
- ✅ 2022**: Explain ledger table compatibility with views (read-only view on ledger)**

7. TRANSACTIONS

   a. Begin Transaction

   b. Commit Transaction

   c. Rollback Transaction

✅ **Include XACT_ABORT for error-based auto-rollback**

✅ **Optional: SAVEPOINT usage in nested transactions**

8. TSQL Programming

   a. Conditional Control Statements

      i. if

      ii. case

      iii. ✅ **2022: DATE_BUCKET() in logic branching (e.g., grouped reports)**

9. Stored Sub Programs

   a. Advantages of Stored Sub Programs compared to Independent SQL Statements

   b. Stored Procedures

      i. Creating, Altering and Dropping

      ii. Optional Parameters

      iii. Input and Output Parameters

c. User Defined Functions
   i. Creating, Altering and Dropping
   ii. Types of User Defined Functions
      1. Scalar Functions

- ✅ **2019: Scalar UDF Inlining optimization**
- ✅ **2022: Use UDFs with improved IQP pipeline (combine with JSON, STRING_AGG())**

2. Table Valued Functions
   a. Inline Table Valued Functions
   b. Multi Statement Table Valued Functions

10. Exception Handling
    a. Implementing Exception Handling
    b. Raising Exceptions Manually

Day 10, 11, 12, 13

11. Query optimization techniques.

12. Indexes
    a. Clustered Index
    b. NonClustered Index
    c. Create, Alter and Drop Indexes
    d. Using Indexes
    e. ✅ **2019:**
       i. **Filtered Indexes, Columnstore Index (intro), Index usage DMVs**
    f. • ✅ **2022:**
       i. **Resumable ALTER INDEX REBUILD**
       ii. **Enhanced Columnstore Index rebuild support**
       iii. **Combine with ledger and in-memory tables (optional advanced topic)**

13. **Working with JSON (New in 2016 → Mature in 2019/2022)**

- **FOR JSON, OPENJSON(), JSON_VALUE(), ISJSON()**
- ✅ **2019: Show practical use in APIs and table joins**
- ✅ **2022: Improved performance, recommended for microservices**