**SQL NOTES**

Delete row: Delete from table\_name where <condition>

**Cascading Referential Integrity**

1. No action (Default option)
2. Cascade
3. NULL
4. Set Default

**Check Constraint**

Delete Constraint: Alter Table table\_name

Drop Constraint constraint\_name

Add Constraint: Alter Table table\_name

Add Constraint constraint\_name Check (Boolean expression)

**Identity Column**

Identity (Seed, Increment)

To reset identity column value,

DBCC CHECKIDENT (table\_name, RESEED, 0)

**Unique Key Constraint**

A table can have only one primary key, but more than one unique key.

Primary key does not allow nulls, where as unique allows one null.

Add unique: Alter table table\_name

Add Constraint constraint\_name Unique (column\_name)

**Retrieving Identity Column**

SCOPE\_IDENTITY () – Same session and same scope

@@IDENTITY () – Same session and across any scope

IDENT\_CURRENT (table\_name) – Specific table across any session and any scope

**Where and Having**

1. Where can be used with Select, Update, insert where as Having can only be used with Select.
2. Where filter rows before aggregation (grouping), where as Having filter groups after aggregations are performed.
3. Aggregate functions can’t be used in where clause where as they can be used in Having clause.

**Replace Null values**

ISNULL () – Returns non-null value

COALESCE () – Return first non-null value

CASE statement

CASE

WHEN expression THEN ‘ ‘ ELSE ‘ ‘

END

**Union and Union All**

1. Union removes duplicate rows, whereas UNION ALL does not
2. Union performs distinct sort to remove duplicates, which makes it slower than Union All.

Union vs Join: Union rows from 2 or more tables, whereas Join combines two or more columns based on logical relationships between table.

**Stored Procedure**

CREATE PROCEDURE OR PROC procedure\_name AS

BEGIN

<body>

END

Run Procedure: procedure\_name, EXEC procedure\_name, EXECUTE procedure\_name

**Advantages of Stored Procedures**

1. Execution plan retention and reusability.
2. Reduces network traffic.
3. Code reusability and maintainability.
4. Better security.
5. Avoids SQL Injection attack.

**String Functions**

ASCII ()

CHAR ()

LTRIM ()

RTRIM ()

LOWER ()

UPPER ()

REVERSE ()

LEN ()

LEFT ()

RIGHT ()

CHARINDEX (expression\_to\_find, expression\_to\_search, [start\_location])

SUBSTRING (expression, start, length)

REPLICATE (string\_to\_be\_replicated, no\_of\_times\_replicate)

SPACE (no\_of\_spaces)

PATINDEX (pattern, expression)

REPLACE (string, pattern\_to\_replace, replace\_pattern)

STUFF (string, start, length, replacement\_expression)

**User Defined Functions (UDF)**

1. Scalar – takes 0 or more values and returns a single (scalar) value.
2. Inline table-valued – returns a table

CREATE FUNCTION function\_name ()

RETURNS TABLE

AS

RETURN (SELECT statement)

1. Multi statement table-valued – returns a table

CREATE FUNCTION function\_name ()

RETURNS @Table Table (table structure)

AS

BEGIN

<body>

END

**Deterministic and Non-Deterministic Functions**

Deterministic functions always return the same result any time they are called with a specific set of input values and given the same state of database. Eg: square, count, power, avg, etc.

Non-Deterministic functions may return different results each time they are called with a specific set of input values even if the database state remains same. Eg: get\_date,

Rand function is both types. When provided seed value it becomes deterministic function.

**SCHEMABINDING**

It specifies that the function is bound to the database objects that it references. When SB is specified, base object can’t be modified in any way that would affect the function definition. The function definition itself be first must be modified or dropped to remove dependencies on the object that is to be modified.

**Temporary Table**

1. Local Temp table
2. Global Temp table

**Indexes**

Clustered Index: is analogous to telephone directory, where the data is arranged by the last name. Table can have one clustered index. However, an index can have multiple columns which is referred as composite clustered index.

Non-clustered Index: is analogous to an index in a textbook. The data is stored in one place, index in another place. The index will have pointers to the storage location. Table can have multiple non-clustered indexes.

Difference b/w clustered and non-clustered Index:

1. Only one clustered index, whereas can have more than one non-clustered index.
2. Clustered index is faster than non-clustered index.
3. Clustered index determines the storage order or rows in table, hence doesn’t require addn. storage space, non-clustered requires addn. space.

**Covering query** – if all the columns that you have requested in the select clause of query, are present in the index, then there is no need to lookup in the table again. The requested data can simply be returned from index.