1. Creating an Incrementing Sequence

To create a sequence object that starts at 1 and increments by 1:

sql Copy code

```sql
CREATE SEQUENCE [dbo].[SequenceObject] AS INT START WITH 1 INCREMENT BY 1;
```

2. Generating the Next Sequence Value

To generate the next sequence value, use the `NEXT VALUE FOR` clause:

sql Copy code

```sql
SELECT NEXT VALUE FOR [dbo].[SequenceObject];
```

Output:
1 *(on the first execution)*
2 *(on the second execution, and so on)*

Each time this query is executed, the sequence value increments by 1. For example, if you execute this query 5 times, the current sequence value will be 5.

3. Retrieving the Current Sequence Value

To view the current sequence value before generating the next value, query the `sys.sequences` table:

sql Copy code

```sql
SELECT name, current_value, increment FROM sys.sequences WHERE name = 'SequenceObject';
```

Output Example:

name	current_value	increment
SequenceObject	5	1

4. Resetting the Sequence Value

To reset the sequence value to start from 1 again:

sql Copy code

```sql
ALTER SEQUENCE [dbo].[SequenceObject] RESTART WITH 1;
```

Verify the reset by generating the next value:

sql Copy code

```sql
SELECT NEXT VALUE FOR [dbo].[SequenceObject];
```

Output: 1

5. Using Sequence Values in an INSERT Query

Example of using sequence values in an `INSERT` statement:

```sql
CREATE TABLE Employees ( Id INT PRIMARY KEY, Name NVARCHAR(50), Gender NVARCHAR(10) ); -- Generate and insert sequence values INSERT INTO Employees VALUES (NEXT VALUE FOR [dbo].[SequenceObject], 'Ben', 'Male'); INSERT INTO Employees VALUES (NEXT VALUE FOR [dbo].[SequenceObject], 'Sara', 'Female'); -- View the data in the table SELECT * FROM Employees;
```

Output Example:

Id	Name	Gender
1	Ben	Male
2	Sara	Female

6. Creating a Decrementing Sequence

To create a sequence object that starts at 100 and decrements by 1:

```sql
CREATE SEQUENCE [dbo].[SequenceObject] AS INT START WITH 100 INCREMENT BY -1;
```

Generate the next value:

```sql
SELECT NEXT VALUE FOR [dbo].[SequenceObject];
```

Output: 100 *(on the first call)*, 99 *(on the second call)*, and so on.

7. Specifying MIN and MAX Values for the Sequence

Use the `MINVALUE` and `MAXVALUE` options to define the range of sequence values.

Step 1: Create the sequence object

```sql
CREATE SEQUENCE [dbo].[SequenceObject] START WITH 100 INCREMENT BY 10 MINVALUE 100 MAXVALUE 150;
```

Step 2: Retrieve the next sequence value

sql Copy code

```sql
SELECT NEXT VALUE FOR [dbo].[SequenceObject];
```

Output:
100 *(on the first call)*, 110 *(on the second call)*, and so on.

When the sequence value reaches 150 (the `MAXVALUE`), you will encounter the following error:

vbnet Copy code

```vbnet
The sequence object 'SequenceObject' has reached its minimum or maximum value. Restart the sequence object to allow new values to be generated.
```

8. Recycling Sequence Values

To restart the sequence from the minimum value after reaching the maximum, enable the `CYCLE` option:

sql Copy code

```sql
ALTER SEQUENCE [dbo].[SequenceObject] INCREMENT BY 10 MINVALUE 100 MAXVALUE 150 CYCLE;
```

After reaching 150, the sequence restarts from 100.

9. Improving Performance with Caching

Sequence object values can be cached to improve performance. Cached values are read from memory instead of the disk.

Example: Create a sequence object with 10 values cached:

sql Copy code

```sql
CREATE SEQUENCE [dbo].[SequenceObject] START WITH 1 INCREMENT BY 1 CACHE 10;
```

When the 11th value is requested, the next 10 values are cached again.

Output Example (Caching in Action):

- **First Request Batch:** Values 1–10 are read from memory.
- **Second Request Batch:** Values 11–20 are cached and used.