🧪 SQL Server Lab: Exploding TempDB with Repeated Inserts

⚠️ WARNING: This lab will intentionally stress TempDB. Only run this in a sandbox environment — not production, QA, or anywhere near a system someone cares about.

# Step 1: Set Up the Environment

1. Use a sandbox SQL Server instance.  
2. Cap the TempDB Size to 5 GB with the following command:

USE [master];  
GO  
ALTER DATABASE [tempdb] MODIFY FILE (  
 NAME = tempdev,  
 MAXSIZE = 5120MB  
);  
GO  
-- Repeat for any additional TempDB files (e.g., tempdev2, tempdev3)

3. Confirm the setting:

SELECT name, size/128.0 AS CurrentSizeMB, max\_size/128.0 AS MaxSizeMB  
FROM tempdb.sys.database\_files;

# Step 2: Create the Big Data Table

Use this to simulate heavy data loads:

USE SandboxDB;  
GO  
  
IF OBJECT\_ID('dbo.LabBigData') IS NOT NULL DROP TABLE dbo.LabBigData;  
  
CREATE TABLE dbo.LabBigData (  
 ID INT IDENTITY(1,1),  
 Name NVARCHAR(100),  
 Value1 INT,  
 Value2 INT,  
 CreatedDate DATETIME DEFAULT GETDATE()  
);  
  
-- Populate with data  
INSERT INTO dbo.LabBigData (Name, Value1, Value2)  
SELECT TOP (100000)   
 'TestName' + CAST(ROW\_NUMBER() OVER (ORDER BY (SELECT NULL)) AS NVARCHAR),  
 ABS(CHECKSUM(NEWID()) % 1000),  
 ABS(CHECKSUM(NEWID()) % 1000)  
FROM sys.all\_objects a CROSS JOIN sys.all\_objects b;

# Step 3: Blow Up the TempDB

Repeated inserts simulate heavy temp table usage:

SET NOCOUNT ON;  
GO  
  
CREATE TABLE #TempExplode (  
 ID INT,  
 Name NVARCHAR(100),  
 Value1 INT,  
 Value2 INT,  
 CreatedDate DATETIME  
);  
  
DECLARE @i INT = 1;  
WHILE @i <= 500  
BEGIN  
 INSERT INTO #TempExplode  
 SELECT \* FROM dbo.LabBigData;  
 SET @i += 1;  
END

## Optional: DROP TABLE in Each Iteration

Does this help free memory?

WHILE @i <= 500  
BEGIN  
 IF OBJECT\_ID('tempdb..#TempExplode') IS NOT NULL DROP TABLE #TempExplode;  
  
 CREATE TABLE #TempExplode (  
 ID INT,  
 Name NVARCHAR(100),  
 Value1 INT,  
 Value2 INT,  
 CreatedDate DATETIME  
 );  
  
 INSERT INTO #TempExplode  
 SELECT \* FROM dbo.LabBigData;  
  
 SET @i += 1;  
END

# Error Messages and Their Meaning

Error 1: TempDB File Size Maxed

Msg 1105: SQL cannot grow the TempDB file beyond maxsize, even if the disk has room.

Error 2: Disk Full

Operating System Error 112: The physical drive is full. SQL cannot allocate more space.

# Wrap-Up Discussion

- Why doesn't TempDB shrink immediately after DROP TABLE?  
- What retains TempDB resources until the session ends?  
- How can multiple users compound TempDB usage?  
- What’s the difference between SQL error 1105 and OS error 112?