# 🧪 SQL Server Temp Table Lab: Training for Controlled Chaos

Objective:  
You're going to learn (the fun way) how temp tables work, how they impact performance, and how to create beautiful chaos in SQL Server that will teach you to respect tempdb. This exercise simulates real-world volume stress, temp table techniques, and performance observation.

## Prerequisite: Download SQLQueryStress

To simulate real-world load, download SQLQueryStress:  
https://github.com/ErikEJ/SqlQueryStress/releases  
It's free, it's glorious, and it helps you abuse your CPU and tempdb with style.

## Part 1: Create the Database (If Needed)

IF DB\_ID('ThePlayPen') IS NULL  
 CREATE DATABASE ThePlayPen;  
GO  
USE ThePlayPen;

## Part 2: Create the Base Table

CREATE TABLE dbo.BigMoodData (  
 ID INT IDENTITY(1,1) PRIMARY KEY,  
 MoodLabel VARCHAR(50),  
 MoodScore INT,  
 CreatedAt DATETIME DEFAULT GETDATE()  
);

## Part 3: Fill It Up - The Fun Way

INSERT INTO dbo.BigMoodData (MoodLabel, MoodScore)  
SELECT TOP 2500  
 CONCAT('Mood\_', ABS(CHECKSUM(NEWID())) % 1000),  
 ABS(CHECKSUM(NEWID())) % 101  
FROM sys.all\_objects a  
CROSS JOIN sys.all\_objects b;

## Part 4: Inflate Like a Pro

-- Run the above INSERT a few more times until you reach ~2.5 million rows.  
-- Optional spicy action:  
DELETE FROM dbo.BigMoodData WHERE ID BETWEEN 2 AND 99;

## Part 5: Add More Moods (Chaos Edition)

WITH RandomMoods AS (  
 SELECT TOP 2500  
 CONCAT('Mood\_', ABS(CHECKSUM(NEWID())) % 1000) AS MoodLabel,  
 ABS(CHECKSUM(NEWID())) % 101 AS MoodScore  
 FROM sys.all\_objects a  
 CROSS JOIN sys.all\_objects b  
)  
INSERT INTO dbo.BigMoodData (MoodLabel, MoodScore)  
SELECT MoodLabel, MoodScore FROM RandomMoods;  
  
WITH ChaosMoods AS (  
 SELECT TOP 1000  
 CASE  
 WHEN ABS(CHECKSUM(NEWID())) % 5 = 0 THEN NULL  
 WHEN ABS(CHECKSUM(NEWID())) % 5 = 1 THEN 'Mood-"Extreme"; DROP TABLES--'  
 WHEN ABS(CHECKSUM(NEWID())) % 5 = 2 THEN 'Unknown???'  
 WHEN ABS(CHECKSUM(NEWID())) % 5 = 3 THEN CHAR(34) + 'Mood: "Deep Feels"' + CHAR(34)  
 ELSE REPLICATE('🤯', ABS(CHECKSUM(NEWID())) % 5 + 1)  
 END AS MoodLabel,  
 CASE  
 WHEN ABS(CHECKSUM(NEWID())) % 4 = 0 THEN NULL  
 WHEN ABS(CHECKSUM(NEWID())) % 4 = 1 THEN -1 \* (ABS(CHECKSUM(NEWID())) % 100)  
 WHEN ABS(CHECKSUM(NEWID())) % 4 = 2 THEN 999999999  
 ELSE CAST(ROUND(RAND(CHECKSUM(NEWID())) \* 100, 3) AS FLOAT)  
 END AS MoodScore  
)  
INSERT INTO dbo.BigMoodData (MoodLabel, MoodScore)  
SELECT MoodLabel, MoodScore FROM ChaosMoods;

## Part 6: Create a Temp Table (Explicit Style)

CREATE TABLE #TempMood1 (  
 ID INT,  
 MoodLabel VARCHAR(50),  
 MoodScore INT,  
 CreatedAt DATETIME,  
 isProcessed BIT  
);  
  
INSERT INTO #TempMood1 (ID, MoodLabel, MoodScore, CreatedAt, isProcessed)  
SELECT ID, MoodLabel, MoodScore, CreatedAt, 0 FROM dbo.BigMoodData;  
  
-- Use SQLQueryStress to simulate multiple concurrent runs and measure performance under load.

## Part 7: Create Another Temp Table (Implicit Style)

SELECT ID, MoodLabel, MoodScore, CreatedAt  
INTO #TempMood2  
FROM dbo.BigMoodData;  
  
ALTER TABLE #TempMood2 ADD isProcessed BIT;  
UPDATE #TempMood2 SET isProcessed = 0;  
  
-- Run multiple threads in SQLQueryStress to compare against Part 6.

## Part 8: Compare Performance

Use SQLQueryStress or Activity Monitor to note the differences:  
- Elapsed time  
- Logical reads  
- CPU seconds  
- Tempdb usage

## Part 9: Monitor Page Usage

-- How bloated is BigMoodData?  
SELECT  
 OBJECT\_NAME(p.object\_id) AS TableName,  
 SUM(p.used\_page\_count) \* 8 AS UsedKB,  
 SUM(p.total\_page\_count) \* 8 AS TotalKB  
FROM sys.dm\_db\_partition\_stats p  
GROUP BY OBJECT\_NAME(p.object\_id);  
  
-- How bloated is tempdb?  
USE tempdb;  
EXEC sp\_spaceused;

## Bonus Round: Break Stuff

ALTER TABLE dbo.BigMoodData  
ADD MoodCategory AS  
 CASE  
 WHEN MoodScore > 80 THEN 'Hype'  
 WHEN MoodScore < 20 THEN 'Trashfire'  
 ELSE 'Mid'  
 END;

## Takeaways for Junior DBAs

- Temp tables are fast but not free  
- SELECT INTO can get you in trouble with schema assumptions  
- Deleting rows causes internal fragmentation  
- Tempdb is a shared resource and easy to abuse  
- Always monitor page count if performance goes sideways

# 📌 Lab Objective

This lab teaches the mechanics and behavior of temp tables in SQL Server under volume stress.  
You'll create realistic workloads and observe how temp tables interact with TempDB, indexes, and concurrent sessions.

# 🧰 Lab Prerequisites

- SQL Server instance with sandbox database permission  
- Ability to create and drop tables  
- Download and install SQLQueryStress: https://github.com/ErikEJ/SqlQueryStress/releases

# 🔧 Environment Setup

If needed, create the lab database and switch to it:

IF DB\_ID('ThePlayPen') IS NULL  
 CREATE DATABASE ThePlayPen;  
GO  
USE ThePlayPen;

# 🧱 Step 1: Create Base Table

Start by building a base table with mood data:

CREATE TABLE dbo.BigMoodData (  
 ID INT IDENTITY(1,1) PRIMARY KEY,  
 MoodLabel VARCHAR(50),  
 MoodScore INT,  
 CreatedAt DATETIME DEFAULT GETDATE()  
);

# 📈 Step 2: Populate It

Insert large amounts of data to simulate a real workload:

INSERT INTO dbo.BigMoodData (MoodLabel, MoodScore)  
SELECT TOP 2500  
 CONCAT('Mood\_', ABS(CHECKSUM(NEWID())) % 1000),  
 ABS(CHECKSUM(NEWID())) % 101  
FROM sys.all\_objects a  
CROSS JOIN sys.all\_objects b;

Repeat the insert multiple times (manual or looped) until you hit ~2.5 million rows.

# 🔥 Step 3: Add Chaotic Data

Introduce variety, nulls, and edge cases:

WITH ChaosMoods AS (  
 SELECT TOP 1000  
 CASE  
 WHEN ABS(CHECKSUM(NEWID())) % 5 = 0 THEN NULL  
 WHEN ABS(CHECKSUM(NEWID())) % 5 = 1 THEN 'Mood-"Extreme"; DROP TABLES--'  
 WHEN ABS(CHECKSUM(NEWID())) % 5 = 2 THEN 'Unknown???'  
 WHEN ABS(CHECKSUM(NEWID())) % 5 = 3 THEN CHAR(34) + 'Mood: "Deep Feels"' + CHAR(34)  
 ELSE REPLICATE('🤯', ABS(CHECKSUM(NEWID())) % 5 + 1)  
 END AS MoodLabel,  
 CASE  
 WHEN ABS(CHECKSUM(NEWID())) % 4 = 0 THEN NULL  
 WHEN ABS(CHECKSUM(NEWID())) % 4 = 1 THEN -1 \* (ABS(CHECKSUM(NEWID())) % 100)  
 WHEN ABS(CHECKSUM(NEWID())) % 4 = 2 THEN 999999999  
 ELSE CAST(ROUND(RAND(CHECKSUM(NEWID())) \* 100, 3) AS FLOAT)  
 END AS MoodScore  
)  
INSERT INTO dbo.BigMoodData (MoodLabel, MoodScore)  
SELECT MoodLabel, MoodScore FROM ChaosMoods;

# 📄 Step 4: Create and Use Temp Tables

Create temp tables using explicit and implicit techniques and insert from your main table.

Then simulate load using SQLQueryStress or manual concurrency.

# 🔍 Step 5: Monitor TempDB

Use these queries to monitor resource usage:

USE tempdb;  
EXEC sp\_spaceused;

# 🧠 Post-Lab Discussion

- What are the memory and performance differences between explicit vs implicit temp table creation?  
- How does repeated use of temp tables affect TempDB?  
- What happens if you DROP a temp table in every iteration?  
- Can you explain SQL Error 1105 vs OS Error 112?

# ✅ Takeaways

- Temp tables are powerful but not free.  
- SELECT INTO has hidden costs and schema limitations.  
- TempDB is a shared resource—don't abuse it blindly.  
- Always monitor performance and space usage when working with temp objects.