**SQL**

**Constraints:**

**Enable/Disable Constraints**

-- Disable all table constraints  
ALTER TABLE YourTableName NOCHECK CONSTRAINT ALL  
-- Enable all table constraints  
ALTER TABLE YourTableName CHECK CONSTRAINT ALL  
-- ----------  
-- Disable single constraint  
ALTER TABLE YourTableName NOCHECK CONSTRAINT YourConstraint  
-- Enable single constraint  
ALTER TABLE YourTableName CHECK CONSTRAINT YourConstraint  
-- ----------  
-- Disable all constraints for database  
EXEC sp\_msforeachtable "ALTER TABLE ? NOCHECK CONSTRAINT all"  
-- Enable all constraints for database  
EXEC sp\_msforeachtable "ALTER TABLE ? WITH CHECK CHECK CONSTRAINT all"

**IDENTITY:**

-- returns the last identity value generated for a specific table in any session and any scope

SELECT IDENT\_CURRENT('TableName');

-- returns the last identity value generated for any table in the current session, across all scopes.

SELECT @@IDENTITY;

-- returns the last identity value generated for any table in the current session and the current scope.

SELECT SCOPE\_IDENTITY();

**SESSION:**

**Kill all user session:**

DECLARE @sqlstring NVARCHAR(max)=''

SELECT @sqlstring = @sqlstring + 'KILL ' + CAST(spid AS VARCHAR(40)) + ';'

FROM sys.sysprocesses

WHERE spid NOT IN (@@spid) -- Exclude current session

AND spid > 50 -- Exclude system spid

PRINT @sqlstring

**SET VS SELECT:**

|  |  |
| --- | --- |
| **SET** | **SELECT** |
| Assign value to a single variable at a time. | Assign a value or to select value from a variable/table/view etc. Assign values to multiple variables. |
| In order to assign values to two different variable two different SET statements are required -- 2 SET statements  SET @s1=1  SET @s2=2 | In this case, the SELECT statement performs better than SET.  -- 1 Select statement  SELECT @s1=1, @s2=2 |
| --Error Handling Scenario, Here the @Error will return as 0 instead of 8134  DECLARE @Error int, @RowCount int  SELECT id/0 FROM dbo.Employee  SET @RowCount = @@ROWCOUNT  SET @Error = @@ERROR  SELECT @Error AS Error  GO | --Correct Error code will get inserted using Select  DECLARE @Error int, @RowCount int  SELECT id/0 FROM dbo.Employee  SELECT @RowCount = @@ROWCOUNT, @Error = @@ERROR  SELECT @Error AS Error |
|  |  |
|  |  |

Triggers:

CREATE TABLE TABLE1

(ID INT, NAME VARCHAR(50), AGE INT)

CREATE TABLE TABLE1\_LOG

(ID INT, NAME VARCHAR(50), AGE INT)

INSERT INTO TABLE1 VALUES (1, 'A', 25)

INSERT INTO TABLE1 VALUES (2, 'B', 26)

--CREATE Trigger for delete.

CREATE TRIGGER TRGRNAME ON TABLE1 FOR DELETE

AS

BEGIN

INSERT INTO TABLE1\_LOG

SELECT \* FROM DELETED

END

--Delete

DELETE FROM TABLE1 WHERE ID = 2

SELECT \* FROM TABLE1\_LOG

**NOCOUNT:**

--It will show "Command(s) completed successfully" .

SET NOCOUNT ON

--it will show "(No. Of row(s) affected)" .

SET NOCOUNT OFF

Set NOCOUNT ON at the top of a proc, before you do any work in the proc, but *also*always

SET NOCOUNT OFF again, before returning any recordsets from the stored proc.

--Used to get the no of rows effected

@@ROWCOUNT