SQL Server Cheat Sheet — Common Scripts & Commands

Generated: 2025-09-13 01:21 UTC

Quick Reference — Data Types

INT, BIGINT, SMALLINT, TINYINT — integers
DECIMAL(p,s), NUMERIC(p,s) — exact precision decimals
FLOAT, REAL — approximate floats
BIT — boolean 0/1
CHAR(n), NCHAR(n) — fixed-length (N* for Unicode)
VARCHAR(n), NVARCHAR(n), NVARCHAR(MAX) — variable-length strings
DATE, TIME, DATETIME, DATETIME2, SMALLDATETIME, DATETIMEOFFSET — date/time types
VARBINARY(MAX), IMAGE (deprecated) — binary data
UNIQUEIDENTIFIER — GUID

Create / Drop / Alter

```
        -- create database
        CREATE DATABASE MyDB;

        -- create table (SQL Server)
        IF NOT EXISTS (SELECT * FROM sysobjects WHERE name='People' AND xtype='U')
        BEGIN
        CREATE TABLE People (
```

Id INT IDENTITY(1,1) PRIMARY KEY,
Name NVARCHAR(200) NOT NULL,
Age INT,
CreatedAt DATETIME DEFAULT GETDATE()
);

```
        END

        -- add column
        ALTER TABLE People ADD Email NVARCHAR(255);

        -- modify column
        ALTER TABLE People ALTER COLUMN Name NVARCHAR(300) NOT NULL;

        -- drop column
        ALTER TABLE People DROP COLUMN Email;

        -- drop table
        DROP TABLE IF EXISTS People;
```

Common DML (SELECT / INSERT / UPDATE / DELETE)

```
        -- select top N
        SELECT TOP 10 * FROM Equipment ORDER BY CreatedAt DESC;

        -- where / like / parameters
        SELECT * FROM Equipment WHERE Machine LIKE '%IMC%';

        -- insert
        INSERT INTO Planner (week_number, year, department, equipment, date)
```

VALUES (37, 2025, 'DC Assembly', 'LINE 17', '2025-09-12');

```
        -- insert and get identity
        INSERT INTO Planner (week_number) VALUES (37);
        SELECT SCOPE_IDENTITY() AS new_id;

        -- update
        UPDATE Planner SET pm_date = '2025-09-19' WHERE id = 123;

        -- delete
        DELETE FROM Planner WHERE id = 123;
```

Pagination / LIMIT

```
        -- SQL Server: OFFSET FETCH (requires ORDER BY)
        SELECT * FROM Equipment
```

ORDER BY EquipmentID

OFFSET 10 ROWS FETCH NEXT 10 ROWS ONLY;

```
-- TOP 1 alternative
SELECT TOP 1 EquipmentID FROM Equipment WHERE Machine = @m ORDER BY EquipmentID;
```

String Functions

LEN(s) -- length
LTRIM(RTRIM(s)) -- trim both sides
UPPER(s), LOWER(s)
SUBSTRING(s, start, length)
CHARINDEX('x', s) -- position
REPLACE(s, 'a', 'b')
CONCAT(a,b,...) -- concatenate
s + t -- string concat (if nullable, consider CONCAT)

Date/Time Functions

GETDATE() -- current date/time (DATETIME)
SYSDATETIME() -- higher precision
GETUTCDATE() -- UTC
DATEADD(day, 7, GETDATE())
DATEDIFF(day, start, end)
FORMAT(date, 'yyyy-MM-dd') -- formatting (slower)
CONVERT(VARCHAR(10), GETDATE(), 120) -- style-based convert

Nulls & Coalesce

IS NULL / IS NOT NULL
COALESCE(a, b, 'default') -- SQL Server
NULLIF(a, b) -- returns NULL if equal

Conditional / CASE / Boolean

CASE WHEN condition THEN value
WHEN ... THEN ...
ELSE other

```
END
```

```
-- example
```

SUM(CASE WHEN LOWER(LTRIM(RTRIM(Alarm_Level))) = 'critical' THEN 1 ELSE 0 END) AS crit

Aggregates & Grouping

COUNT(*), COUNT(DISTINCT col), SUM(col), AVG(col), MIN(col), MAX(col)
GROUP BY col1, col2
HAVING SUM(col) > 0

Window Functions

ROW_NUMBER() OVER (ORDER BY CreatedAt DESC) AS rn
RANK(), DENSE_RANK()
SUM(value) OVER (PARTITION BY category ORDER BY date ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
LEAD(), LAG()

Joins

INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN, CROSS JOIN

```
-- Example
SELECT e.*, t.Test_Date
```

FROM Equipment e
LEFT JOIN CBM_Testing t ON t.Equipment_ID = e.EquipmentID

Transactions

```
BEGIN TRANSACTION;
-- statements
```
COMMIT TRANSACTION;
```
-- or rollback
```
ROLLBACK TRANSACTION;

```
-- check transaction state
IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION;
```

Indexes

```
-- create index
CREATE INDEX IX_Equipment_Machine ON Equipment(Machine);

-- unique index
CREATE UNIQUE INDEX UX_People_Email ON People(Email);

-- drop index
DROP INDEX IX_Equipment_Machine ON Equipment;
```

Constraints

```
-- primary key, foreign key, unique, check
ALTER TABLE Orders ADD CONSTRAINT FK_Orders_Customers FOREIGN KEY (CustomerId) REFERENCES
Customers(Id);
ALTER TABLE Products ADD CONSTRAINT CHK_Price_Positive CHECK (Price >= 0);
```

Stored Procedures & Parameters

```
-- create stored procedure
CREATE PROCEDURE dbo.GetEquipmentByMachine
```
@machine NVARCHAR(200)
AS
```
BEGIN
SET NOCOUNT ON;
SELECT * FROM Equipment WHERE Machine = @machine;
END

-- execute
EXEC dbo.GetEquipmentByMachine @machine = 'IMC04';
```

Temporary Tables & Table Variables

```
-- temp table (session)
CREATE TABLE #temp (id INT, name NVARCHAR(100));
INSERT INTO #temp VALUES (1, 'a');
SELECT * FROM #temp;

-- table variable
DECLARE @t TABLE (id INT, name NVARCHAR(100));
INSERT INTO @t VALUES (1, 'a');
SELECT * FROM @t;
```

Error handling / Try-Catch

```
BEGIN TRY
-- statements
END TRY
BEGIN CATCH
SELECT ERROR_NUMBER() AS ErrNo, ERROR_MESSAGE() AS ErrMsg;
-- optional: ROLLBACK TRANSACTION;
END CATCH
```

Permissions & Users

```
-- create login and user
CREATE LOGIN app_user WITH PASSWORD = 'P@ssw0rd';
```

USE CBM;
```
CREATE USER app_user FOR LOGIN app_user;
EXEC sp_addrolemember 'db_datareader', 'app_user';
EXEC sp_addrolemember 'db_datawriter', 'app_user';
```

Backup & Restore

```
-- backup
BACKUP DATABASE CBM TO DISK = 'C:\backups\CBM.bak' WITH FORMAT;

-- restore (careful, may require MOVE)
RESTORE DATABASE CBM FROM DISK = 'C:\backups\CBM.bak' WITH REPLACE;
```

Admin / Server Info

```
-- list databases
SELECT name, state_desc FROM sys.databases;

-- server version
SELECT @@VERSION;

-- active connections
SELECT DB_NAME(dbid) AS DBName, COUNT(spid) AS Connections FROM sys.sysprocesses GROUP BY dbid;
```

Common Migration Fixes from SQLite

```
-- AUTOINCREMENT -> IDENTITY(1,1)
-- last_insert_rowid() -> SCOPE_IDENTITY()
-- sqlite_master -> INFORMATION_SCHEMA.TABLES / sys.objects
-- PRAGMA table_info(table) -> INFORMATION_SCHEMA.COLUMNS
-- TRIM(x) -> LTRIM(RTRIM(x))
-- datetime('now') -> GETDATE()
-- LIMIT -> OFFSET/FETCH or TOP
```

Performance Tips

```
-- use proper indexing (covering indexes)
-- avoid SELECT *
-- use SET NOCOUNT ON in stored procs
-- filter early (WHERE) and avoid functions on indexed columns
-- update statistics: UPDATE STATISTICS dbo.MyTable;
-- check execution plan (SSMS Query -> Include Actual Execution Plan)
```

Useful Built-in Functions

SCOPE_IDENTITY(), @@IDENTITY (use SCOPE_IDENTITY), ISNULL(), COALESCE()
TRY_CONVERT(), TRY_CAST()
JSON_VALUE(), JSON_QUERY() -- JSON functions
OPENJSON to parse JSON

Quick Troubleshooting

```
-- Check user permissions
SELECT permission_name FROM fn_my_permissions(NULL, 'DATABASE');

-- Check table existence
SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'Planner';

-- Check columns
SELECT COLUMN_NAME, DATA_TYPE FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = 'Planner';
```

Contacts / Tools

```
-- Helpful tools: SQL Server Management Studio (SSMS), Azure Data Studio, SQL Server Migration
Assistant (SSMA)
-- Drivers: ODBC Driver 17/18 for SQL Server, Microsoft.Data.SqlClient for .NET
```