Microsoft SQL Server Cheatsheet

A comprehensive cheat sheet for Microsoft SQL Server, covering essential commands, syntax, and functions for database management and querying.

Basic SQL Commands

Data Definition Language (DDL)

CREATE DATABASE	Creates a new database. `CREATE DATABASE MyDatabase;`
ALTER DATABASE	Modifies an existing database. `ALTER DATABASE MyDatabase MODIFY NAME = MyNewDatabase;`
DROP DATABASE	Deletes a database. `DROP DATABASE MyDatabase;`

CREATE TABLE	Creates a new table.
	`CREATE TABLE Employees (`
	`    ID INT PRIMARY KEY,`
	`    Name VARCHAR(255)`
	`);`

ALTER TABLE	Modifies an existing table. `ALTER TABLE Employees ADD Salary DECIMAL(10, 2);`
DROP TABLE	Deletes a table. `DROP TABLE Employees;`

Data Manipulation Language (DML)

SELECT	Retrieves data from a database. `SELECT * FROM Employees;`
INSERT	Inserts new data into a table. `INSERT INTO Employees (ID, Name) VALUES (1, 'John Doe');`
UPDATE	Updates existing data in a table. `UPDATE Employees SET Salary = 50000 WHERE ID = 1;`
DELETE	Deletes data from a table. `DELETE FROM Employees WHERE ID = 1;`

MERGE	Performs insert, update, or delete operations based on conditions.
	`MERGE INTO TargetTable AS Target`
	`USING SourceTable AS Source`
	`ON Target.ID = Source.ID`
	`WHEN MATCHED THEN`
	`    UPDATE SET Target.Name = Source.Name`
	`WHEN NOT MATCHED THEN`
	`    INSERT (ID, Name) VALUES (Source.ID, Source.Name);`

Querying Data

Filtering and Sorting

WHERE	Filters rows based on a condition. `SELECT * FROM Employees WHERE Salary > 60000;`
AND / OR	Combines multiple conditions. `SELECT * FROM Employees WHERE Salary > 50000 AND Department = 'IT';`
ORDER BY	Sorts the result set. `SELECT * FROM Employees ORDER BY Name ASC;`
TOP	Returns the top N rows. `SELECT TOP 10 * FROM Employees ORDER BY Salary DESC;`
BETWEEN	Filters rows within a range. `SELECT * FROM Employees WHERE Salary BETWEEN 50000 AND 70000;`
IN	Filters rows based on a set of values. `SELECT * FROM Employees WHERE Department IN ('IT', 'HR');`

Joins

INNER JOIN	Returns rows with matching values in both tables. `SELECT * FROM Employees INNER JOIN Departments ON Employees.DepartmentID = Departments.ID;`
LEFT JOIN	Returns all rows from the left table and matching rows from the right table. `SELECT * FROM Employees LEFT JOIN Departments ON Employees.DepartmentID = Departments.ID;`
RIGHT JOIN	Returns all rows from the right table and matching rows from the left table. `SELECT * FROM Employees RIGHT JOIN Departments ON Employees.DepartmentID = Departments.ID;`
FULL OUTER JOIN	Returns all rows when there is a match in either the left or right table. `SELECT * FROM Employees FULL OUTER JOIN Departments ON Employees.DepartmentID = Departments.ID;`
CROSS JOIN	Returns the Cartesian product of the tables. `SELECT * FROM Employees CROSS JOIN Departments;`

Advanced SQL Features

Aggregate Functions

COUNT	Counts the number of rows. `SELECT COUNT(*) FROM Employees;`
SUM	Calculates the sum of values. `SELECT SUM(Salary) FROM Employees;`
AVG	Calculates the average of values. `SELECT AVG(Salary) FROM Employees;`
MIN	Finds the minimum value. `SELECT MIN(Salary) FROM Employees;`
MAX	Finds the maximum value. `SELECT MAX(Salary) FROM Employees;`

Grouping and Having

GROUP BY	Groups rows with the same values. `SELECT Department, COUNT(*) FROM Employees GROUP BY Department;`
HAVING	Filters groups based on a condition. `SELECT Department, COUNT(*) FROM Employees GROUP BY Department HAVING COUNT(*) > 10;`
ROLLUP	Generates multiple grouping sets, including subtotals and grand totals. `SELECT Department, YEAR(HireDate), COUNT(*) FROM Employees GROUP BY ROLLUP (Department, YEAR(HireDate));`
CUBE	Generates all possible grouping sets for the specified columns. `SELECT Department, YEAR(HireDate), COUNT(*) FROM Employees GROUP BY CUBE (Department, YEAR(HireDate));`

Subqueries

Subquery in WHERE clause	Using a subquery to filter results. `SELECT * FROM Employees WHERE DepartmentID IN (SELECT ID FROM Departments WHERE Location = 'New York');`
Subquery in SELECT clause	Using a subquery to return a value. `SELECT Name, (SELECT MAX(Salary) FROM Employees) AS MaxSalary FROM Employees;`
Correlated Subquery	A subquery that references a column from the outer query. `SELECT Name FROM Employees e1 WHERE Salary > (SELECT AVG(Salary) FROM Employees e2 WHERE e1.DepartmentID = e2.DepartmentID);`

Transactions and Stored Procedures

Transactions

BEGIN TRANSACTION	Starts a new transaction. `BEGIN TRANSACTION;`
COMMIT TRANSACTION	Saves all changes made during the transaction. `COMMIT TRANSACTION;`
ROLLBACK TRANSACTION	Reverts all changes made during the transaction. `ROLLBACK TRANSACTION;`
SAVE TRANSACTION	Sets a savepoint within a transaction. `SAVE TRANSACTION SavePoint1;`

Stored Procedures

CREATE PROCEDURE	Creates a new stored procedure. `CREATE PROCEDURE GetEmployeesByDepartment (@Department VARCHAR(255)) AS BEGIN SELECT * FROM Employees WHERE Department = @Department; END;`
EXECUTE PROCEDURE	Executes a stored procedure. `EXEC GetEmployeesByDepartment 'IT';`
ALTER PROCEDURE	Modifies an existing stored procedure. `ALTER PROCEDURE GetEmployeesByDepartment (@Department VARCHAR(255)) AS BEGIN SELECT ID, Name FROM Employees WHERE Department = @Department; END;`
DROP PROCEDURE	Deletes a stored procedure. `DROP PROCEDURE GetEmployeesByDepartment;`