

# SQL-Spickzettel

Dieser Spickzettel fasst die wichtigsten SQL-Grundbefehle und deren Reihenfolge zusammen. Er eignet sich ideal zur Prüfungsvorbereitung (AP1 / AP2) und als tägliche Referenz.

## Grundstruktur

Die Standard-Struktur einer SQL-Abfrage folgt immer derselben Reihenfolge:

[snippet.sql](#)

```
SELECT ...  
FROM ...  
WHERE ...  
GROUP BY ...  
HAVING ...  
ORDER BY ...  
LIMIT ...;
```

**Merke:** Wenn du die Struktur immer vollständig aufschreibst, kannst du danach Schritt für Schritt alles ergänzen und Unnötiges streichen.

## Schlüsselbefehle im Überblick

Schlüsselwort	Bedeutung	Beispiel
<b>SELECT</b>	Welche Spalten angezeigt werden	SELECT name, preis
<b>FROM</b>	Aus welcher Tabelle die Daten kommen	FROM produkte
<b>WHERE</b>	Filtert Zeilen <b>vor</b> der Gruppierung	WHERE lagerbestand > 100
<b>GROUP BY</b>	Gruppiert gleiche Werte	GROUP BY kategorie
<b>HAVING</b>	Filtert <b>nach</b> der Gruppierung	HAVING AVG(preis) > 50
<b>ORDER BY</b>	Sortiert die Ausgabe	ORDER BY preis DESC
<b>LIMIT</b>	Begrenzt die Zeilenanzahl	LIMIT 10

## Beispiel 1 - Einfache Abfrage

### Aufgabe:

Namen und Preise aller Produkte mit Lagerbestand > 100, nach Preis absteigend sortiert, max. 10 Einträge.

### snippet.sql

```
SELECT name, preis
FROM produkte
WHERE lagerbestand > 100
ORDER BY preis DESC
LIMIT 10;
```

#### Erklärung:

- WHERE filtert nur Zeilen mit Lagerbestand über 100.
  - ORDER BY preis DESC sortiert die Ausgabe absteigend.
  - LIMIT 10 zeigt nur die ersten 10 Ergebnisse an.
- 

## Beispiel 2 - Gruppierte Auswertung

#### Aufgabe:

Pro Kategorie den Durchschnittspreis für Produkte mit Lagerbestand > 100 berechnen. Nur Kategorien mit einem Durchschnittspreis > 50 anzeigen, absteigend sortiert, max. 10 Ergebnisse.

### snippet.sql

```
SELECT kategorie, AVG(preis) AS durchschnittspreis
FROM produkte
WHERE lagerbestand > 100
GROUP BY kategorie
HAVING AVG(preis) > 50
ORDER BY durchschnittspreis DESC
LIMIT 10;
```

#### Erklärung:

- AVG(preis) berechnet den Durchschnittspreis je Kategorie.
  - GROUP BY kategorie fasst Produkte derselben Kategorie zusammen.
  - HAVING filtert nur Gruppen mit Durchschnitt > 50.
  - Das Alias durchschnittspreis kann in ORDER BY wiederverwendet werden.
- 

## Unterschied: WHERE vs. HAVING

Vergleichspunkt	WHERE	HAVING
Zeitpunkt	Vor der Gruppierung	Nach der Gruppierung

<b>Filtert</b>	Einzelne Datensätze	Gruppenergebnisse
<b>Beispiel</b>	WHERE preis > 50	HAVING AVG(preis) > 50

**Merksatz:**

„WHERE prüft Datensätze, HAVING prüft Gruppen.“

## Wichtige Aggregatfunktionen

Funktion	Beschreibung	Beispiel
<b>COUNT()</b>	Zählt Datensätze	COUNT(*)
<b>SUM()</b>	Summiert Werte	SUM(preis)
<b>AVG()</b>	Durchschnitt	AVG(preis)
<b>MIN()</b>	Kleinster Wert	MIN(preis)
<b>MAX()</b>	Größter Wert	MAX(preis)

**Hinweis:**

Aggregatfunktionen kannst du nur in Verbindung mit GROUP BY oder zur Gesamtauswertung nutzen.

## Sortierung

[snippet.sql](#)

```
ORDER BY spalte [ASC|DESC]
```

Parameter	Bedeutung
<b>ASC</b>	Aufsteigend (Standard)
<b>DESC</b>	Absteigend

Beispiel:

[snippet.sql](#)

```
ORDER BY preis DESC, name ASC;
```

→ Sortiert zuerst nach Preis (absteigend), dann nach Name (aufsteigend).

## LIMIT & OFFSET

Befehl	Bedeutung
-----	-----
LIMIT n	Zeigt nur n Zeilen an
OFFSET n	Überspringt n Zeilen vor der Ausgabe

Beispiel:

[snippet.sql](#)

```
SELECT * FROM produkte
ORDER BY preis DESC
LIMIT 10 OFFSET 20;
```

→ Zeigt die Zeilen 21-30 der Sortierung.

## Aliasnamen

Mit AS kannst du Spalten- oder Tabellennamen umbenennen:

[snippet.sql](#)

```
SELECT name AS produktname, preis AS einzelpreis
FROM produkte AS p;
```

Aliases erleichtern lesbare Ergebnisse und sind in ORDER BY wiederverwendbar.

## Joins (Überblick)

Join-Typ	Beschreibung	Beispiel
-----	-----	-----
<b>INNER JOIN</b>	Nur passende Datensätze beider Tabellen	SELECT * FROM kunden INNER JOIN bestellungen ON kunden.id = bestellungen.kunden_id;
<b>LEFT JOIN</b>	Alle Datensätze links + passende rechts	SELECT * FROM kunden LEFT JOIN bestellungen ON kunden.id = bestellungen.kunden_id;
<b>RIGHT JOIN</b>	Alle Datensätze rechts + passende links	SELECT * FROM kunden RIGHT JOIN bestellungen ON kunden.id = bestellungen.kunden_id;

<b>FULL JOIN</b>	Alle Datensätze beider Seiten (falls unterstützt)	SELECT * FROM kunden FULL JOIN bestellungen ON ...;
------------------	---	---

### Merksatz:

„INNER = nur Treffer, LEFT = alles links + Treffer, RIGHT = alles rechts + Treffer.“

## Nützliche Prüfungsbefehle

[snippet.sql](#)

```
-- Zeilen zählen
SELECT COUNT(*) FROM produkte;

-- Alle Kategorien ohne Wiederholung
SELECT DISTINCT kategorie FROM produkte;

-- Höchster Preis pro Kategorie
SELECT kategorie, MAX(preis) FROM produkte GROUP BY kategorie;

-- Preisaktualisierung
UPDATE produkte SET preis = preis * 1.1 WHERE kategorie = 'Hardware';

-- Neue Zeile einfügen
INSERT INTO produkte (name, preis, lagerbestand) VALUES ('Monitor', 199, 50);

-- Datensatz löschen
DELETE FROM produkte WHERE name = 'Altgerät';
```

## Mini-Merksätze

- „SELECT zeigt, FROM liefert, WHERE filtert.“
- „GROUP BY fasst, HAVING prüft, ORDER BY sortiert.“
- „COUNT zählt, AVG mittelt, SUM addiert.“
- „Ohne WHERE = alles.“
- „Immer mit Semikolon abschließen.“

## Prüfungstipp

SQL in Ruhe logisch lesen:

1. FROM – welche Tabelle(n)?
  2. WHERE – welche Bedingungen?
  3. GROUP BY – wie gruppieren?
  4. HAVING – was bleibt übrig?
  5. ORDER BY – wie sortieren?
  6. LIMIT – wie viele Zeilen?
- 

From: <http://wiki.nctl.de/dokuwiki/> - ☐ **Veni. Vidi. sudo rm -rf / vici.**

Permanent link: [http://wiki.nctl.de/dokuwiki/doku.php?id=it-themen:allgemein:sql\\_spickzettel&rev=1759745264](http://wiki.nctl.de/dokuwiki/doku.php?id=it-themen:allgemein:sql_spickzettel&rev=1759745264)

Last update: **06.10.2025 12:07**

