# SQL Cheat-Sheet – Quick Reference

> **Kompakte Referenz** basierend auf Lectures 7-10 **Themen:** SELECT, DDL/DML, Normalisierung, Joins

---

## 📖 Wie nutze ich dieses Cheat-Sheet?

Dieses Dokument ist eine **schnelle Referenz** für die wichtigsten SQL-Konzepte aus den Lectures 7-10:

- **L8:** CREATE TABLE, ALTER, DROP, INSERT, UPDATE, DELETE, Constraints
- **L7:** SELECT, WHERE, ORDER BY, GROUP BY, Aggregation
- **L9:** Normalisierung (1NF, 2NF, 3NF), ER-Diagramme
- **L10:** Joins (INNER, LEFT, RIGHT, FULL, CROSS), Subqueries

**Navigation:**

- Nutze das **Inhaltsverzeichnis** (☰ links oben)
- Springe direkt zum gewünschten Thema
- Alle Code-Beispiele sind **interaktiv** – probiere sie aus!

---

## 1. DDL – Tabellen definieren (L8)

### 1.1 CREATE TABLE

```sql
-- Grundstruktur
CREATE TABLE customers (
  customer_id INTEGER PRIMARY KEY,
  first_name TEXT NOT NULL,
  last_name TEXT NOT NULL,
  email TEXT UNIQUE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```
-- Grundstruktur
CREATE TABLE customers (
  customer_id INTEGER PRIMARY KEY,
  first_name TEXT NOT NULL,
  last_name TEXT NOT NULL,
  email TEXT UNIQUE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
)
```

*ok*

## 1.2 Datentypen

| Typ | Verwendung | Beispiel |
|---|---|---|
| `INTEGER` | Ganzzahlen | `age INTEGER` |
| `BIGINT` | Große Ganzzahlen | `user_id BIGINT` |
| `DECIMAL(p,s)` | Präzise Dezimalzahlen | `price DECIMAL(10,2)` |
| `FLOAT/DOUBLE` | Approximative Zahlen | `latitude FLOAT` |
| `TEXT` | Unbegrenzter Text | `description TEXT` |
| `VARCHAR(n)` | Text mit Längenlimit | `email VARCHAR(255)` |
| `CHAR(n)` | Festlängen-Text | `country_code CHAR(2)` |
| `DATE` | Datum | `birth_date DATE` |
| `TIMESTAMP` | Datum + Uhrzeit | `created_at TIMESTAMP` |
| `BOOLEAN` | Wahr/Falsch | `is_active BOOLEAN` |
| `JSON` | JSON-Objekte | `metadata JSON` |

## 1.3 Constraints

```
1 CREATE TABLE orders (
2   order_id INTEGER PRIMARY KEY,                    -- Eindeutig + NOT
3   customer_id INTEGER NOT NULL,                    -- Pflichtfeld
4   total DECIMAL(10,2) CHECK (total >= 0)           -- Validierung
```

```
  total DECIMAL(10,2) CHECK (total >= 0),                    -- Validierung
  status TEXT CHECK (status IN ('pending', 'shipped', 'delivered')),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  -- Standardwert
  FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
```

```
CREATE TABLE orders (
  order_id INTEGER PRIMARY KEY,                    -- Eindeutig + NOT
NULL
  customer_id INTEGER NOT NULL,                    -- Pflichtfeld
  total DECIMAL(10,2) CHECK (total >= 0),          -- Validierung
  status TEXT CHECK (status IN ('pending', 'shipped', 'delivered')),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  -- Standardwert
  FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
)
```

*ok*

**Constraint-Übersicht:**

| Constraint | Zweck | Beispiel |
|---|---|---|
| `PRIMARY KEY` | Eindeutig + NOT NULL | `user_id INTEGER PRIMARY KEY` |
| `FOREIGN KEY` | Referenzielle Integrität | `FOREIGN KEY (user_id) REFERENCES users(user_id)` |
| `UNIQUE` | Eindeutig (NULL erlaubt) | `email TEXT UNIQUE` |
| `NOT NULL` | Pflichtfeld | `name TEXT NOT NULL` |
| `CHECK` | Benutzerdefiniert | `age INTEGER CHECK (age >= 18)` |
| `DEFAULT` | Standardwert | `status TEXT DEFAULT 'active'` |

## 1.4 Foreign Keys

```
-- Mit ON DELETE / ON UPDATE
CREATE TABLE order_items (
  item_id INTEGER PRIMARY KEY,
  order_id INTEGER,
  product_id INTEGER,
  quantity INTEGER,
```

```
  FOREIGN KEY (order_id) REFERENCES orders(order_id)
    ON DELETE CASCADE,          -- Löscht Items, wenn Order gelöscht
  FOREIGN KEY (product_id) REFERENCES products(product_id)
    ON DELETE RESTRICT          -- Verhindert Löschen, wenn Items
      existieren
);
```

```
-- Mit ON DELETE / ON UPDATE
CREATE TABLE order_items (
  item_id INTEGER PRIMARY KEY,
  order_id INTEGER,
  product_id INTEGER,
  quantity INTEGER,
  FOREIGN KEY (order_id) REFERENCES orders(order_id)
    ON DELETE CASCADE,          -- Löscht Items, wenn Order gelöscht
wird
  FOREIGN KEY (product_id) REFERENCES products(product_id)
    ON DELETE RESTRICT          -- Verhindert Löschen, wenn Items
existieren
)
```

```
relation "products" does not exist
```

**ON DELETE Optionen:**

| Option | Verhalten |
|---|---|
| `CASCADE` | Abhängige Zeilen werden gelöscht |
| `SET NULL` | Foreign Key wird auf NULL gesetzt |
| `RESTRICT` | Löschen wird verhindert (Standard) |
| `NO ACTION` | Wie RESTRICT |

## 1.5 ALTER TABLE

```
-- Spalte hinzufügen
ALTER TABLE customers
ADD COLUMN phone TEXT;

-- Spalte ändern (nicht alle DBs unterstützen alle Operationen)
ALTER TABLE customers
ALTER COLUMN email SET NOT NULL;

```

```sql
-- Spalte umbenennen (PostgreSQL)
ALTER TABLE customers
RENAME COLUMN phone TO mobile;

-- Tabelle umbenennen
ALTER TABLE customers RENAME TO clients;

-- Spalte löschen
ALTER TABLE clients
DROP COLUMN mobile;
```

```sql
-- Spalte hinzufügen
ALTER TABLE customers
ADD COLUMN phone TEXT
```

*ok*

```sql
-- Spalte ändern (nicht alle DBs unterstützen alle Operationen)
ALTER TABLE customers
ALTER COLUMN email SET NOT NULL
```

*ok*

```sql
-- Spalte umbenennen (PostgreSQL)
ALTER TABLE customers
RENAME COLUMN phone TO mobile
```

*ok*

```sql
-- Tabelle umbenennen
ALTER TABLE customers RENAME TO clients
```

*ok*

```sql
-- Spalte löschen
ALTER TABLE clients
DROP COLUMN mobile
```

*ok*

## 1.6 DROP TABLE

```sql
-- Einfach
DROP TABLE IF EXISTS temp_table;

-- Mit CASCADE (löscht auch abhängige Objekte)
DROP TABLE products CASCADE;

-- Mit RESTRICT (verhindert Löschen bei Abhängigkeiten)
```

```
8  DROP TABLE products RESTRICT;
```

```
-- Einfach
DROP TABLE IF EXISTS temp_table
ok

-- Mit CASCADE (löscht auch abhängige Objekte)
DROP TABLE products CASCADE
table "products" does not exist
```

## 2. DML – Daten manipulieren (L8)

### 2.1 INSERT

```
1  -- Einzelne Zeile
2  INSERT INTO customers (customer_id, first_name, last_name, email)
3  VALUES (1, 'Alice', 'Anderson', 'alice@email.com');
4
5  -- Mehrere Zeilen (Bulk Insert)
6  INSERT INTO customers (customer_id, first_name, last_name, email) VAL
7    (2, 'Bob', 'Brown', 'bob@email.com'),
8    (3, 'Carol', 'Clark', 'carol@email.com'),
9    (4, 'David', 'Davis', 'david@email.com');
10
11 -- INSERT ... SELECT (Daten kopieren)
12 INSERT INTO customers_backup (customer_id, first_name, last_name, ema
13 SELECT customer_id, first_name, last_name, email
14 FROM customers
15 WHERE email IS NOT NULL;
16
17 -- INSERT ... ON CONFLICT (Upsert)
18 INSERT INTO customers (customer_id, first_name, last_name, email)
19 VALUES (1, 'Alice Updated', 'alice_new@email.com')
20 ON CONFLICT (customer_id) DO UPDATE
21 SET
22   first_name = EXCLUDED.first_name,
23   email = EXCLUDED.email;
```

```
-- Einzelne Zeile
INSERT INTO customers (customer_id, first_name, last_name, email)
VALUES (1, 'Alice', 'Anderson', 'alice@email.com')
```

```
relation "customers" does not exist
```

## 2.2 UPDATE

```sql
-- Einzelne Zeile
UPDATE customers
SET email = 'alice_new@email.com'
WHERE customer_id = 1;

-- Mehrere Spalten
UPDATE customers
SET
  first_name = 'Alice',
  last_name = 'Smith',
  email = 'alice.smith@email.com'
WHERE customer_id = 1;

-- Berechnete Updates
UPDATE products
SET price = price * 1.10;  -- 10% Preiserhöhung

-- UPDATE mit Subquery
UPDATE products
SET price = price * (1 - (
  SELECT discount_percent / 100
  FROM categories
  WHERE categories.category_id = products.category_id
));
```

```
-- Einzelne Zeile
UPDATE customers
SET email = 'alice_new@email.com'
WHERE customer_id = 1
```

```
relation "customers" does not exist
```

⚠️ **Immer WHERE nutzen!** Ohne WHERE werden ALLE Zeilen geändert!

## 2.3 DELETE

```sql
-- Einzelne Zeile
```

```sql
DELETE FROM customers
WHERE customer_id = 5;

-- Mehrere Zeilen
DELETE FROM customers
WHERE email IS NULL;

-- DELETE mit Subquery
DELETE FROM customers
WHERE customer_id NOT IN (
  SELECT DISTINCT customer_id FROM orders
);
```

```
-- Einzelne Zeile
DELETE FROM customers
WHERE customer_id = 5
```

```
relation "customers" does not exist
```

⚠️ **Immer WHERE nutzen!** Ohne WHERE werden ALLE Zeilen gelöscht!

## 2.4 TRUNCATE vs DELETE

```sql
-- DELETE: Selektiv, in Transaktion reversibel
DELETE FROM products WHERE price < 50;

-- TRUNCATE: Löscht alle Zeilen, schneller, meist nicht reversibel
TRUNCATE TABLE products;
```

```
-- DELETE: Selektiv, in Transaktion reversibel
DELETE FROM products WHERE price < 50
```

```
relation "products" does not exist
```

| Feature | DELETE | TRUNCATE |
|---|---|---|
| WHERE-Klausel | ✅ Ja | ❌ Nein |
| Performance | Langsamer | Schneller |
| Rollback | ✅ Möglich | ⚠️ DB-abhängig |
| Triggers | ✅ Werden ausgelöst | ❌ Meist nicht |
| Auto-Increment | Bleibt | Wird zurückgesetzt |

---

## 3. SELECT – Daten abfragen (L7)

### 3.1 Grundstruktur

```
SELECT spalte1, spalte2, ...
FROM tabelle
WHERE bedingung
ORDER BY spalte [ASC|DESC]
LIMIT anzahl;
```

### 3.2 Beispieldatenbank laden

```
INSTALL httpfs;
LOAD httpfs;

CREATE TEMP TABLE products_raw AS
SELECT *
FROM read_json('https://raw.githubusercontent.com/andre-dietrich
  /Datenbankensysteme-Vorlesung/refs/heads/main/assets/dat/products.j

CREATE TABLE products (
  product_id INTEGER PRIMARY KEY,
  name TEXT,
  category TEXT,
  brand TEXT,
  price REAL,
  stock INTEGER,
  rating REAL,
  created_at DATE
);

INSERT INTO products
```

```sql
SELECT
  CAST(product_id AS INTEGER),
  name::TEXT,
  category::TEXT,
  brand::TEXT,
  CAST(price AS REAL),
  CAST(stock AS INTEGER),
  CAST(rating AS REAL),
  CAST(created_at AS DATE)
FROM products_raw;
```

```
INSTALL httpfs
```

**No data**

```
LOAD httpfs
```

**No data**

```
CREATE TEMP TABLE products_raw AS
SELECT *
FROM read_json('https://raw.githubusercontent.com/andre-
dietrich/Datenbankensysteme-
Vorlesung/refs/heads/main/assets/dat/products.json')
```

| | Count |
|---|---|
| *1* | 932 |

```
CREATE TABLE products (
  product_id INTEGER PRIMARY KEY,
  name TEXT,
  category TEXT,
  brand TEXT,
  price REAL,
  stock INTEGER,
  rating REAL,
  created_at DATE
)
```

**No data**

```
INSERT INTO products
SELECT
  CAST(product_id AS INTEGER),
  name::TEXT,
  category::TEXT,
  brand::TEXT,
  CAST(price AS REAL),
  CAST(stock AS INTEGER),
  CAST(rating AS REAL),
  CAST(created_at AS DATE)
FROM products_raw
```

| | Count |
|---|---|
| 1 | 932 |

## 3.3 Spalten auswählen

```sql
-- Alle Spalten
SELECT * FROM products LIMIT 5;

-- Spezifische Spalten
SELECT name, price, category FROM products LIMIT 5;

-- Mit Alias
SELECT
  name AS product_name,
  price AS price_eur,
  category
FROM products LIMIT 5;
```

```
Catalog Error: Table with name products does not exist!
Did you mean "pg_proc"?
LINE 2: SELECT * FROM products LIMIT 5;
                      ^
```

## 3.4 WHERE – Filtern

```sql
-- Vergleichsoperatoren: =, <>, <, >, <=, >=
SELECT name, price FROM products
WHERE price < 50;

-- Logische Operatoren: AND, OR, NOT
SELECT name, price, category FROM products
WHERE category = 'Electronics' AND price < 100;

-- BETWEEN
SELECT name, price FROM products
WHERE price BETWEEN 100 AND 500;

-- IN
SELECT name, category FROM products
WHERE category IN ('Electronics', 'Office', 'Groceries');

-- LIKE (Pattern Matching)
SELECT name FROM products
WHERE name LIKE '%Laptop%';  -- % = beliebig viele Zeichen
```

```sql

-- NULL-Werte
SELECT name, rating FROM products
WHERE rating IS NULL;
```

| | name | price |
|---|---|---|
| 1 | 4K Monitor | 24.920000076293945 |
| 2 | Basic T-Shirt Max | 24.34000015258789 |
| 3 | Jacket | 44 |
| 4 | Jacket L | 29.280000686645508 |
| 5 | Raincoat | 43.810001373291016 |
| 6 | Jeans - Silver | 34.15999984741211 |
| 7 | Leggings L | 18.100000381469727 |
| 8 | Jeans | 35.779998779296875 |
| 9 | Cap Lite | 35.54999923706055 |
| 10 | Dress Shirt - Blue | 39.970001220703125 |
| 11 | Chinos | 8.640000343322754 |
| 12 | Leggings L | 11.569999694824219 |
| 13 | Polo Shirt Pro XL | 14.930000305175781 |
| 14 | Sneakers | 9.789999961853027 |
| 15 | Leggings | 31.280000686645508 |
| 16 | Hoodie Lite L - White | 30.030000686645508 |
| 17 | Jeans Max | 12.520000457763672 |
| 18 | Dress Shirt 2.0 S | 6.130000114440918 |
| 19 | Hoodie | 34.900001525878906 |
| 20 | Leggings | 40.04999923706055 |
| 21 | Chinos Max M | 38.029998779296875 |
| 22 | Jacket Plus M | 40.52000045776367 |
| 23 | Granola 750g | 31.600000381469727 |
| 24 | Olive Oil 1L - Silver | 26.729999542236328 |
| 25 | Tomato Sauce 400g Max XL | 34.119998931884766 |
| 26 | Olive Oil 1L M | 36.099998474121094 |
| 27 | Green Tea 50 bags M | 26.450000762939453 |
| 28 | Green Tea 50 bags - Black | 20.040000915527344 |
| 29 | Arabica Coffee Beans 1kg | 0.8999999761581421 |
| 30 | Pasta Fusilli 500g Plus | 38.16999816894531 |
| 31 | Olive Oil 1L M - Green | 18.639999389648438 |

| | | |
|---|---|---|
| 32 | Green Tea 50 bags Max | 35.939998626708984 |
| 33 | Arabica Coffee Beans 1kg Plus | 19.8799991607666 |
| 34 | Sea Salt 500g | 19.399999618530273 |
| 35 | Green Tea 50 bags | 38.52000045776367 |
| 36 | Arabica Coffee Beans 1kg | 37.90999984741211 |
| 37 | Pasta Fusilli 500g Max M | 4.489999771118164 |
| 38 | Basmati Rice 1kg L - White | 18.389999389648438 |
| 39 | Tomato Sauce 400g | 10.350000381469727 |
| 40 | Dark Chocolate 85% 100g XL - Green | 30.329999923706055 |
| 41 | Tomato Sauce 400g | 29.100000381469727 |
| 42 | Basmati Rice 1kg Plus - White | 33.790000915527344 |
| 43 | Basmati Rice 1kg Max L | 4.760000228881836 |
| 44 | Basmati Rice 1kg Lite S | 38.29999923706055 |
| 45 | Granola 750g Plus - Green | 16.09000015258789 |
| 46 | Organic Oat Milk 1L | 19.229999542236328 |
| 47 | Dark Chocolate 85% 100g | 37.93000030517578 |
| 48 | Sea Salt 500g Max - Blue | 29.719999313354492 |
| 49 | Tomato Sauce 400g - Black | 5.71999979019165 |
| 50 | Olive Oil 1L | 29.25 |
| 51 | Organic Oat Milk 1L | 6.03000020980835 |
| 52 | Green Tea 50 bags - Silver | 23.989999771118164 |
| 53 | Honey 500g 2.0 - White | 34.7599983215332 |
| 54 | Sea Salt 500g - White | 34.369998931884766 |
| 55 | Olive Oil 1L | 34.540000915527344 |
| 56 | Green Tea 50 bags S - Red | 8.079999923706055 |
| 57 | Organic Oat Milk 1L | 26.049999237060547 |
| 58 | Arabica Coffee Beans 1kg Pro XL - Blue | 33.56999969482422 |
| 59 | Honey 500g 2.0 M | 13.140000343322754 |
| 60 | Olive Oil 1L M | 15.430000305175781 |
| 61 | Arabica Coffee Beans 1kg Lite | 21.65999984741211 |
| 62 | Granola 750g Max | 6.110000133514404 |
| 63 | Basmati Rice 1kg Pro | 4.389999866485596 |

| | | |
|---|---|---|
| 64 | Almonds 200g Lite XL | 2.4800000190734863 |
| 65 | Pasta Fusilli 500g Max XL | 7.199999809265137 |
| 66 | Basmati Rice 1kg | 14.489999771118164 |
| 67 | Organic Oat Milk 1L | 13.609999656677246 |
| 68 | Sea Salt 500g | 38.40999984741211 |
| 69 | Dark Chocolate 85% 100g Lite L - Green | 23.610000610351562 |
| 70 | Pasta Fusilli 500g Lite M | 13.270000457763672 |
| 71 | Sea Salt 500g Lite - Graphite | 3.069999933242798 |
| 72 | Organic Oat Milk 1L Lite S - Red | 2.25 |
| 73 | Honey 500g Lite S | 8.130000114440918 |
| 74 | Tomato Sauce 400g | 23.399999618530273 |
| 75 | Basmati Rice 1kg 2.0 - Red | 5.320000171661377 |
| 76 | Green Tea 50 bags Plus | 39.84000015258789 |
| 77 | Almonds 200g - Green | 31.09000015258789 |
| 78 | Granola 750g XL - Graphite | 1.9900000095367432 |
| 79 | Green Tea 50 bags | 29.260000228881836 |
| 80 | Tomato Sauce 400g - White | 27.360000610351562 |
| 81 | Pasta Fusilli 500g - Black | 32.2599983215332 |
| 82 | Arabica Coffee Beans 1kg - Green | 18.68000030517578 |
| 83 | Sea Salt 500g - Green | 13.100000381469727 |
| 84 | Green Tea 50 bags XL | 15.170000076293945 |
| 85 | Honey 500g - Black | 1.7300000190734863 |
| 86 | Almonds 200g | 3.5899999141693115 |
| 87 | Granola 750g - White | 30.3799991607666 |
| 88 | Dark Chocolate 85% 100g - White | 15.75 |
| 89 | Dark Chocolate 85% 100g - White | 22.690000534057617 |
| 90 | Pasta Fusilli 500g - Silver | 38.970001220703125 |
| 91 | Tomato Sauce 400g L - Black | 16.030000686645508 |
| 92 | Sea Salt 500g | 26.670000076293945 |
| 93 | Almonds 200g Max | 26.280000686645508 |
| 94 | Olive Oil 1L - Silver | 5.409999847412109 |
| 95 | Tomato Sauce 400g Max | 10.800000190734863 |

| | | |
|---|---|---|
| *96* | Almonds 200g - Graphite | 28.170000076293945 |
| *97* | Granola 750g Plus | 32.529998779296875 |
| *98* | Granola 750g | 39.65999984741211 |
| *99* | Almonds 200g Max - White | 7.199999809265137 |
| *100* | Sea Salt 500g - White | 24.15999984741211 |
| *101* | Honey 500g | 13.029999732971191 |
| *102* | Arabica Coffee Beans 1kg | 3.0899999141693115 |
| *103* | Honey 500g Max | 34.66999816894531 |
| *104* | Arabica Coffee Beans 1kg | 36.58000183105469 |
| *105* | Dark Chocolate 85% 100g L - Red | 20.34000015258789 |
| *106* | Organic Oat Milk 1L Max | 5.329999923706055 |
| *107* | Green Tea 50 bags | 3.680000066757202 |
| *108* | Green Tea 50 bags | 32.880001068115234 |
| *109* | Sea Salt 500g | 16.290000915527344 |
| *110* | Organic Oat Milk 1L 2.0 | 6.860000133514404 |
| *111* | Green Tea 50 bags | 38.06999969482422 |
| *112* | Honey 500g Max S | 36.369998931884766 |
| *113* | Honey 500g - Black | 28.209999084472656 |
| *114* | Basmati Rice 1kg | 2.3299999237060547 |
| *115* | Granola 750g 2.0 - Blue | 37.529998779296875 |
| *116* | Dark Chocolate 85% 100g M - Black | 28.280000686645508 |
| *117* | Sea Salt 500g Max XL - Silver | 37.43000030517578 |
| *118* | Green Tea 50 bags - Black | 17.030000686645508 |
| *119* | Arabica Coffee Beans 1kg Plus - Blue | 17.31999969482422 |
| *120* | Organic Oat Milk 1L Plus M | 15.989999771118164 |
| *121* | Arabica Coffee Beans 1kg Max XL - Silver | 22.3799991607666 |
| *122* | Dark Chocolate 85% 100g Max S - Silver | 10.670000076293945 |
| *123* | Office Scissors | 13.710000038146973 |
| *124* | Sticky Notes (12 pads) Lite | 11.960000038146973 |
| *125* | Ergonomic Chair Plus | 15.399999618530273 |
| *126* | A4 Copy Paper 500 sheets | 33.79999923706055 |
| *127* | Ergonomic Chair | 11.920000076293945 |

| | | |
|---|---|---|
| *128* | Whiteboard Markers (8-pack) Max - Blue | 33.04999923706055 |
| *129* | Whiteboard Markers (8-pack) | 30.940000534057617 |
| *130* | Gel Pens (10-pack) Lite | 10.020000457763672 |
| *131* | Binder 5cm | 23.209999084472656 |
| *132* | Laptop Stand Lite | 7.53000020980835 |
| *133* | Air Fryer 4L | 27.290000915527344 |
| *134* | Toaster 2-slice Lite | 32.65999984741211 |
| *135* | Cutlery Set (24 pcs) Lite - White | 15.220000267028809 |
| *136* | Remote Control Car Plus L | 12.180000305175781 |
| *137* | Science Kit - Volcano XL | 7.480000019073486 |
| *138* | Plush Bear 30cm - Silver | 20.1299991607666 |
| *139* | Remote Control Car XL - Green | 30.920000076293945 |
| *140* | Marble Run 2.0 | 6.849999904632568 |
| *141* | RC Drone Mini S | 43.599998474121094 |
| *142* | Plush Bear 30cm | 24.15999984741211 |
| *143* | Building Blocks Set Max - Blue | 27.860000610351562 |
| *144* | Remote Control Car Plus XL | 26.950000762939453 |
| *145* | STEM Robot Kit | 28.110000610351562 |
| *146* | Building Blocks Set L | 21.34000015258789 |
| *147* | RC Drone Mini Plus | 40.650001525878906 |
| *148* | RC Drone Mini | 41.189998626708984 |
| *149* | Wooden Train Set S - Blue | 29.25 |
| *150* | Building Blocks Set Lite | 44.119998931884766 |
| *151* | Building Blocks Set Max | 46.45000076293945 |
| *152* | Puzzle 1000 pieces Pro - Blue | 7.699999809265137 |
| *153* | Puzzle 1000 pieces Pro L - White | 36.349998474121094 |
| *154* | Remote Control Car Pro | 30.729999542236328 |
| *155* | Wooden Train Set M | 37.599998474121094 |
| *156* | Bike Helmet L | 32.13999938964844 |
| *157* | Basketball Max M | 40.439998626708984 |
| *158* | Compression Shirt | 43.619998931884766 |
| *159* | Bike Helmet | 24.809999465942383 |

| | | |
|---|---|---|
| *160* | Tennis Racket Pro | 42.029998779296875 |
| *161* | Dumbbells Pair 2x5kg | 44.63999938964844 |
| *162* | Running Shoes 2.0 S | 37.63999938964844 |
| *163* | Football Size 5 Plus S | 31.270000457763672 |
| *164* | Cycling Gloves Pro - Blue | 30.440000534057617 |
| *165* | Moisturizing Cream 50ml L - Black | 41.900001525878906 |
| *166* | Face Serum 30ml Max | 31.690000534057617 |
| *167* | Hair Oil 100ml - Green | 12.15999984741211 |
| *168* | Lip Balm Plus XL | 5.570000171661377 |
| *169* | Face Serum 30ml XL | 39.400001525878906 |
| *170* | Body Lotion 250ml Pro S - Graphite | 4.489999771118164 |
| *171* | Lip Balm Max - Black | 38.77000045776367 |
| *172* | Conditioner 300ml 2.0 | 47.0099983215332 |
| *173* | Face Mask Sheet XL | 9.920000076293945 |
| *174* | Moisturizing Cream 50ml Max | 31.440000534057617 |
| *175* | Body Lotion 250ml | 25.170000076293945 |
| *176* | Hair Oil 100ml | 30.290000915527344 |
| *177* | Face Serum 30ml Plus M | 29.360000610351562 |
| *178* | Micellar Water 400ml Max XL | 29.65999984741211 |
| *179* | Moisturizing Cream 50ml Pro - Green | 48.63999938964844 |
| *180* | Conditioner 300ml XL | 32.34000015258789 |
| *181* | Facial Cleanser 200ml Max | 49.540000915527344 |
| *182* | Face Mask Sheet | 22.299999237060547 |
| *183* | Lip Balm Max | 30.829999923706055 |
| *184* | Facial Cleanser 200ml | 31.850000381469727 |
| *185* | Body Lotion 250ml | 16.459999084472656 |
| *186* | Facial Cleanser 200ml | 19.850000381469727 |
| *187* | Conditioner 300ml L | 38.5099983215332 |
| *188* | Lip Balm | 24.459999084472656 |
| *189* | Face Mask Sheet | 16.059999465942383 |
| *190* | Sunscreen SPF50 100ml | 39.84000015258789 |
| *191* | Facial Cleanser 200ml S | 46.029998779296875 |

| | | |
|---|---|---|
| *192* | Paperback Novel Lite M | 16.389999389648438 |
| *193* | Poetry Collection Pro - White | 27.969999313354492 |
| *194* | Poetry Collection Pro | 42.52000045776367 |
| *195* | Poetry Collection - Green | 29.3799991607666 |
| *196* | Cookbook - Quick Meals Plus | 15.600000381469727 |
| *197* | Short Stories Plus | 44.93000030517578 |
| *198* | Short Stories Max S - Blue | 9.479999542236328 |
| *199* | Travel Guide: Japan Max | 17.829999923706055 |
| *200* | History of Art M - Green | 37.959999084472656 |
| *201* | Business Strategy 101 - Red | 9.579999923706055 |
| *202* | Beginners German Lite | 48.86000061035156 |
| *203* | History of Art Pro | 13.640000343322754 |
| *204* | Cookbook - Quick Meals XL | 30.979999542236328 |
| *205* | Short Stories L - Graphite | 13.130000114440918 |
| *206* | Paperback Novel Pro S | 44.72999954223633 |
| *207* | Poetry Collection M | 19.6200008392334 |
| *208* | Short Stories - Graphite | 17.450000762939453 |
| *209* | Beginners German Lite | 29.290000915527344 |
| *210* | Travel Guide: Japan Pro - Graphite | 6.190000057220459 |
| *211* | Cookbook - Quick Meals Lite | 36.86000061035156 |
| *212* | Data Science Handbook | 37.349998474121094 |
| *213* | Travel Guide: Japan | 16.190000534057617 |
| *214* | Poetry Collection - Silver | 7.059999942779541 |
| *215* | Travel Guide: Japan Plus - Black | 42.15999984741211 |
| *216* | Short Stories Max M | 33.40999984741211 |
| *217* | Travel Guide: Japan 2.0 | 39.20000076293945 |
| *218* | Business Strategy 101 Pro M | 25.579999923706055 |
| *219* | Data Science Handbook | 22.209999084472656 |
| *220* | Data Science Handbook | 49.91999816894531 |
| *221* | Business Strategy 101 XL - Blue | 26.110000610351562 |
| *222* | Travel Guide: Japan M - White | 12.729999542236328 |
| *223* | Business Strategy 101 - Green | 27.65999984741211 |

| | | |
|---|---|---|
| *224* | Poetry Collection XL | 30.1200008392334 |
| *225* | Data Science Handbook | 47.290000915527344 |
| *226* | Business Strategy 101 - Silver | 22.309999465942383 |
| *227* | History of Art XL | 22.030000686645508 |
| *228* | Data Science Handbook Pro | 7.800000190734863 |
| *229* | Paperback Novel Pro S | 48.380001068115234 |
| *230* | Short Stories - Black | 11.819999694824219 |
| *231* | History of Art | 31.450000762939453 |
| *232* | Data Science Handbook L | 29.709999084472656 |
| *233* | Childrens Stories M | 15.600000381469727 |
| *234* | Poetry Collection - Green | 26.690000534057617 |
| *235* | Mystery Thriller - Silver | 38.540000915527344 |
| *236* | Paperback Novel | 23.780000686645508 |
| *237* | Beginners German | 46.689998626708984 |
| *238* | Mystery Thriller Lite | 47.529998779296875 |
| *239* | Short Stories | 21.190000534057617 |
| *240* | History of Art - Silver | 28.010000228881836 |
| *241* | Childrens Stories 2.0 S | 11.010000228881836 |
| *242* | Fantasy Epic Vol. 1 | 27.100000381469727 |
| *243* | History of Art | 36.849998474121094 |
| *244* | Data Science Handbook S | 28.540000915527344 |
| *245* | Childrens Stories Plus | 43.09000015258789 |
| *246* | Travel Guide: Japan | 19.700000762939453 |
| *247* | Poetry Collection Max S | 28.489999771118164 |
| *248* | Data Science Handbook | 28.219999313354492 |
| *249* | Beginners German XL | 22.43000030517578 |
| *250* | Paperback Novel - Blue | 6.539999961853027 |
| *251* | Travel Guide: Japan XL | 48.869998931884766 |
| *252* | Pet Treats Mixed 500g Lite XL - Red | 19.700000762939453 |
| *253* | Bird Seed 1kg 2.0 - Blue | 23.3799991607666 |
| *254* | Pet Treats Mixed 500g Pro | 13.420000076293945 |
| *255* | Dog Leash Nylon Plus - Green | 21.559999465942383 |

| | | |
|---|---|---|
| 256 | Aquarium Filter | 48.619998931884766 |
| 257 | Bird Seed 1kg | 13.520000457763672 |
| 258 | Aquarium Filter | 29.239999771118164 |
| 259 | Pet Treats Mixed 500g M | 10.529999732971191 |
| 260 | Pet Shampoo 250ml Pro - Red | 26.75 |
| 261 | Pet Shampoo 250ml XL - Black | 38.939998626708984 |
| 262 | Pet Bed Medium Max XL | 42.11000061035156 |
| 263 | Pet Bed Medium Plus | 3.950000047683716 |
| 264 | Bird Seed 1kg Pro S - Silver | 4.980000019073486 |
| 265 | Pet Bed Medium | 18.260000228881836 |
| 266 | Dog Leash Nylon - Silver | 17.139999389648438 |
| 267 | Aquarium Filter XL | 47.83000183105469 |
| 268 | Aquarium Filter - Blue | 5.789999961853027 |
| 269 | Pet Shampoo 250ml M | 45.09000015258789 |
| 270 | Fish Flakes 200g Max XL | 31.030000686645508 |
| 271 | Small Animal Hay 1kg L - Silver | 32.189998626708984 |
| 272 | Cat Scratching Post Plus | 5.829999923706055 |
| 273 | Bird Seed 1kg | 4.300000190734863 |

## 3.5 ORDER BY – Sortieren

```sql
-- Aufsteigend (Standard)
SELECT name, price FROM products
ORDER BY price ASC
LIMIT 10;

-- Absteigend
SELECT name, price FROM products
ORDER BY price DESC
LIMIT 10;

-- Nach mehreren Spalten
SELECT category, name, price FROM products
ORDER BY category ASC, price DESC
LIMIT 10;

-- NULL-Behandlung
SELECT name, rating FROM products
```

```
ORDER BY rating DESC NULLS LAST
LIMIT 10;
```

| | name | price |
|---|---|---|
| 1 | Arabica Coffee Beans 1kg | 0.8999999761581421 |
| 2 | Honey 500g - Black | 1.7300000190734863 |
| 3 | Granola 750g XL - Graphite | 1.9900000095367432 |
| 4 | Organic Oat Milk 1L Lite S - Red | 2.25 |
| 5 | Basmati Rice 1kg | 2.3299999237060547 |
| 6 | Almonds 200g Lite XL | 2.4800000190734863 |
| 7 | Sea Salt 500g Lite - Graphite | 3.069999933242798 |
| 8 | Arabica Coffee Beans 1kg | 3.0899999141693115 |
| 9 | Almonds 200g | 3.5899999141693115 |
| 10 | Green Tea 50 bags | 3.680000066757202 |

## 3.6 DISTINCT – Duplikate entfernen

```sql
-- Eindeutige Kategorien
SELECT DISTINCT category
FROM products
ORDER BY category;

-- Eindeutige Kombinationen
SELECT DISTINCT category, brand
FROM products
ORDER BY category, brand;
```

| | category |
|---|---|
| 1 | Beauty |
| 2 | Books |
| 3 | Clothing |
| 4 | Electronics |
| 5 | Groceries |
| 6 | Home & Kitchen |
| 7 | Office |
| 8 | Pets |
| 9 | Sports |
| 10 | Toys |

## 3.7 GROUP BY & Aggregation

```sql
-- Anzahl pro Kategorie
SELECT
  category,
  COUNT(*) AS product_count
FROM products
GROUP BY category
ORDER BY product_count DESC;

-- Mehrere Aggregate
SELECT
  category,
  COUNT(*) AS products,
  ROUND(AVG(price), 2) AS avg_price,
  MIN(price) AS cheapest,
  MAX(price) AS most_expensive,
  SUM(stock) AS total_stock
FROM products
GROUP BY category;

-- HAVING - Gruppen filtern
SELECT
  category,
  COUNT(*) AS count,
  ROUND(AVG(price), 2) AS avg_price
FROM products
GROUP BY category
HAVING COUNT(*) > 50
```

```sql
ORDER BY count DESC;

-- WHERE + GROUP BY + HAVING
SELECT
  brand,
  COUNT(*) AS products,
  ROUND(AVG(price), 2) AS avg_price
FROM products
WHERE category = 'Electronics'
GROUP BY brand
HAVING COUNT(*) > 5
ORDER BY products DESC;
```

| | category | product_count |
|---|---|---|
| 1 | Electronics | 100 |
| 2 | Groceries | 100 |
| 3 | Pets | 98 |
| 4 | Books | 95 |
| 5 | Office | 94 |
| 6 | Toys | 93 |
| 7 | Clothing | 92 |
| 8 | Sports | 91 |
| 9 | Home & Kitchen | 90 |
| 10 | Beauty | 79 |

## 3.8 LIMIT & OFFSET – Paginierung

```sql
-- Top 10
SELECT name, price FROM products
ORDER BY price DESC
LIMIT 10;

-- Seite 2 (Zeilen 11-20)
SELECT name, price FROM products
ORDER BY name
LIMIT 10 OFFSET 10;

-- Cursor-basierte Paginierung (besser bei großen Offsets)
SELECT product_id, name, price FROM products
WHERE product_id > 'P00010'
ORDER BY product_id
```

```
ORDER BY product_id
LIMIT 10;
```

| | name | price |
|---|---|---|
| 1 | Bluetooth Speaker | 1493.8199462890625 |
| 2 | Mechanical Keyboard | 1462.4000244140625 |
| 3 | 4K Monitor | 1445.8599853515625 |
| 4 | Tablet 10 | 1433.699951171875 |
| 5 | Soundbar Pro S | 1425.739990234375 |
| 6 | Bluetooth Speaker Lite M | 1425.0899658203125 |
| 7 | Gaming Mouse | 1402.1300048828125 |
| 8 | Laptop 14" 2.0 | 1399.030029296875 |
| 9 | 4K Monitor - White | 1390.68994140625 |
| 10 | 4K Monitor XL | 1389.72998046875 |

# 4. Normalisierung (L9)

## 4.1 Normalformen-Übersicht

| Normalform | Regel | Beispiel-Problem |
|---|---|---|
| **1NF** | Alle Werte sind atomar (keine Listen) | Adresse = „Straße, PLZ, Stadt“ |
| **2NF** | 1NF + keine partiellen Abhängigkeiten | Produktname hängt nur von product_id ab |
| **3NF** | 2NF + keine transitiven Abhängigkeiten | Stadt hängt von PLZ ab |

## 4.2 1NF – Erste Normalform

**Regel:** Alle Attribute sind atomar (kein einziges Attribut enthält mehrere Werte).

```
-- ❌ Nicht 1NF (Adresse nicht atomar)
CREATE TABLE users_bad (
  user_id INTEGER,
```

```
  name TEXT,
  address TEXT  -- "Hauptstraße 12, 04109 Leipzig"
);

-- ✅ 1NF (Adresse aufgeteilt)
CREATE TABLE users_good (
  user_id INTEGER,
  name TEXT,
  street TEXT,      -- "Hauptstraße 12"
  zip TEXT,         -- "04109"
  city TEXT         -- "Leipzig"
);
```

```
-- ❌ Nicht 1NF (Adresse nicht atomar)
CREATE TABLE users_bad (
  user_id INTEGER,
  name TEXT,
  address TEXT  -- "Hauptstraße 12, 04109 Leipzig"
)
```

*ok*

```
-- ✅ 1NF (Adresse aufgeteilt)
CREATE TABLE users_good (
  user_id INTEGER,
  name TEXT,
  street TEXT,      -- "Hauptstraße 12"
  zip TEXT,         -- "04109"
  city TEXT         -- "Leipzig"
)
```

*ok*

## 4.3 2NF – Zweite Normalform

**Regel:** 1NF + jedes Nicht-Schlüssel-Attribut hängt vom gesamten Primärschlüssel ab (keine partiellen Abhängigkeiten).

```
-- ❌ Nicht 2NF (product_name hängt nur von product_id ab, nicht von order_id)
CREATE TABLE order_items_bad (
  order_id INTEGER,
  product_id INTEGER,
  product_name TEXT,    -- Redundant! Hängt nur von product_id ab
  quantity INTEGER,
  PRIMARY KEY (order_id, product_id)
);
```

```sql
-- ✅ 2NF (Produktdaten ausgelagert)
CREATE TABLE products (
  product_id INTEGER PRIMARY KEY,
  product_name TEXT
);

CREATE TABLE order_items_good (
  order_id INTEGER,
  product_id INTEGER,
  quantity INTEGER,
  PRIMARY KEY (order_id, product_id),
  FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```

```sql
-- ❌ Nicht 2NF (product_name hängt nur von product_id ab, nicht von order_id)
CREATE TABLE order_items_bad (
  order_id INTEGER,
  product_id INTEGER,
  product_name TEXT,    -- Redundant! Hängt nur von product_id ab
  quantity INTEGER,
  PRIMARY KEY (order_id, product_id)
)
```

*ok*

```sql
-- ✅ 2NF (Produktdaten ausgelagert)
CREATE TABLE products (
  product_id INTEGER PRIMARY KEY,
  product_name TEXT
)
```

*ok*

```sql
CREATE TABLE order_items_good (
  order_id INTEGER,
  product_id INTEGER,
  quantity INTEGER,
  PRIMARY KEY (order_id, product_id),
  FOREIGN KEY (product_id) REFERENCES products(product_id)
)
```

*ok*

## 4.4 3NF – Dritte Normalform

**Regel:** 2NF + keine transitiven Abhängigkeiten (kein Nicht-Schlüssel-Attribut hängt von einem anderen Nicht-Schlüssel-Attribut ab).

```sql
-- ❌ Nicht 3NF (city hängt von zip ab → transitive Abhängigkeit)
CREATE TABLE customers_bad (
  customer_id INTEGER PRIMARY KEY,
  name TEXT,
  zip TEXT,
  city TEXT  -- Hängt von zip ab!
);

-- ✅ 3NF (PLZ/Stadt ausgelagert)
CREATE TABLE locations (
  location_id INTEGER PRIMARY KEY,
  zip TEXT UNIQUE,
  city TEXT
);

CREATE TABLE customers_good (
  customer_id INTEGER PRIMARY KEY,
  name TEXT,
  location_id INTEGER,
  FOREIGN KEY (location_id) REFERENCES locations(location_id)
);
```


```sql
-- ❌ Nicht 3NF (city hängt von zip ab → transitive Abhängigkeit)
CREATE TABLE customers_bad (
  customer_id INTEGER PRIMARY KEY,
  name TEXT,
  zip TEXT,
  city TEXT  -- Hängt von zip ab!
)
```


*ok*


```sql
-- ✅ 3NF (PLZ/Stadt ausgelagert)
CREATE TABLE locations (
  location_id INTEGER PRIMARY KEY,
  zip TEXT UNIQUE,
  city TEXT
)
```


*ok*


```sql
CREATE TABLE customers_good (
  customer_id INTEGER PRIMARY KEY,
  name TEXT,
  location_id INTEGER,
  FOREIGN KEY (location_id) REFERENCES locations(location_id)
)
```


*ok*

## 4.5 ER-Diagramm-Notation

**Kardinalitäten:**

- **1:1** (Eins-zu-Eins): `User ||--|| Profile` – Ein User hat genau ein Profil
- **1:N** (Eins-zu-Viele): `User ||--o{ Order` – Ein User kann viele Orders haben
- **N:M** (Viele-zu-Viele): `Product }o--o{ Order` – Viele Products in vielen Orders (via Junction Table)

**Schlüssel:**

- **PK** = Primary Key (unterstrichen)
- **FK** = Foreign Key (gestrichelt)

---

# 5. Joins (L10)

## 5.1 Testdaten erstellen

```sql
-- Locations
CREATE TABLE locations (
  location_id INTEGER PRIMARY KEY,
  city TEXT NOT NULL,
  postal_code TEXT NOT NULL
);

INSERT INTO locations VALUES
  (1, 'Berlin', '10115'),
  (2, 'Hamburg', '20095'),
  (3, 'Munich', '80331');

-- Customers
CREATE TABLE customers (
  customer_id INTEGER PRIMARY KEY,
  first_name TEXT NOT NULL,
  last_name TEXT NOT NULL,
  location_id INTEGER,
  FOREIGN KEY (location_id) REFERENCES locations(location_id)
);

INSERT INTO customers VALUES
  (1, 'Alice', 'Anderson', 1),
  (2, 'Bob', 'Brown', 2),
  (3, 'Carol', 'Clark', 3),
  (4, 'David', 'Davis', NULL);  -- Kein Standort

-- Orders
CREATE TABLE orders (
```

```
  order_id INTEGER PRIMARY KEY,
  customer_id INTEGER,
  order_date DATE,
  FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);

INSERT INTO orders VALUES
  (101, 1, '2024-01-15'),
  (102, 1, '2024-02-20'),
  (103, 2, '2024-01-22'),
  (104, 3, '2024-03-10');
-- Kunde 4 hat keine Bestellung!
```

```
-- Locations
CREATE TABLE locations (
  location_id INTEGER PRIMARY KEY,
  city TEXT NOT NULL,
  postal_code TEXT NOT NULL
)
```

*ok*

```
INSERT INTO locations VALUES
  (1, 'Berlin', '10115'),
  (2, 'Hamburg', '20095'),
  (3, 'Munich', '80331')
```

*ok*

```
-- Customers
CREATE TABLE customers (
  customer_id INTEGER PRIMARY KEY,
  first_name TEXT NOT NULL,
  last_name TEXT NOT NULL,
  location_id INTEGER,
  FOREIGN KEY (location_id) REFERENCES locations(location_id)
)
```

*ok*

```
INSERT INTO customers VALUES
  (1, 'Alice', 'Anderson', 1),
  (2, 'Bob', 'Brown', 2),
  (3, 'Carol', 'Clark', 3),
  (4, 'David', 'Davis', NULL)
```

*ok*

```
-- Kein Standort

-- Orders
CREATE TABLE orders (
  order_id INTEGER PRIMARY KEY,
  customer_id INTEGER,
  order_date DATE,
  FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
)
```

*ok*

```
INSERT INTO orders VALUES
  (101, 1, '2024-01-15'),
  (102, 1, '2024-02-20'),
```

```
  (103, 2, '2024-01-22'),
  (104, 3, '2024-03-10')
ok

-- Kunde 4 hat keine Bestellung!
ok
```

## 5.2 INNER JOIN

**Regel:** Nur Zeilen, die in BEIDEN Tabellen übereinstimmen.

```sql
SELECT
  c.first_name,
  c.last_name,
  o.order_id,
  o.order_date
FROM customers c
INNER JOIN orders o ON c.customer_id = o.customer_id;
```

```
SELECT
  c.first_name,
  c.last_name,
  o.order_id,
  o.order_date
FROM customers c
INNER JOIN orders o ON c.customer_id = o.customer_id
```

| # | first_name | last_name | order_id | order_date |
|---|---|---|---|---|
| 1 | Alice | Anderson | 101 | 2024-01-15 |
| 2 | Alice | Anderson | 102 | 2024-02-20 |
| 3 | Bob | Brown | 103 | 2024-01-22 |
| 4 | Carol | Clark | 104 | 2024-03-10 |

4 rows

**Ergebnis:** Nur Kunden MIT Bestellungen (Alice, Bob, Carol). David fehlt!

## 5.3 LEFT JOIN (LEFT OUTER JOIN)

**Regel:** Alle Zeilen aus der linken Tabelle + passende aus der rechten. Wenn keine Übereinstimmung → NULL.

```sql
SELECT
```

```
SELECT
  c.first_name,
  c.last_name,
  o.order_id,
  o.order_date
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id;
```

```
SELECT
  c.first_name,
  c.last_name,
  o.order_id,
  o.order_date
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id
```

| # | first_name | last_name | order_id | order_date |
|---|---|---|---|---|
| 1 | Alice | Anderson | 101 | 2024-01-15 |
| 2 | Alice | Anderson | 102 | 2024-02-20 |
| 3 | Bob | Brown | 103 | 2024-01-22 |
| 4 | Carol | Clark | 104 | 2024-03-10 |
| 5 | David | Davis | *null* | *null* |

5 rows

**Ergebnis:** Alle Kunden (auch David), aber bei David ist order_id = NULL.

**Use Case:** „Zeige alle Kunden, auch die ohne Bestellung."

## 5.4 RIGHT JOIN (RIGHT OUTER JOIN)

**Regel:** Alle Zeilen aus der rechten Tabelle + passende aus der linken.

```
SELECT
  c.first_name,
  o.order_id
FROM customers c
RIGHT JOIN orders o ON c.customer_id = o.customer_id;
```

```
SELECT
  c.first_name,
  o.order_id
FROM customers c
RIGHT JOIN orders o ON c.customer_id = o.customer_id
```

| # | first_name | order_id |
|---|---|---|
| 1 | Alice | 101 |
| 2 | Alice | 102 |
| 3 | Bob | 103 |
| 4 | Carol | 104 |

4 rows

**In der Praxis:** Selten genutzt (kann durch LEFT JOIN ersetzt werden).

## 5.5 FULL OUTER JOIN

**Regel:** Alle Zeilen aus BEIDEN Tabellen. Wenn keine Übereinstimmung → NULL.

```sql
SELECT
  c.first_name,
  o.order_id
FROM customers c
FULL OUTER JOIN orders o ON c.customer_id = o.customer_id;
```

```
SELECT
  c.first_name,
  o.order_id
FROM customers c
FULL OUTER JOIN orders o ON c.customer_id = o.customer_id
```

| # | first_name | order_id |
|---|---|---|
| 1 | Alice | 101 |
| 2 | Alice | 102 |
| 3 | Bob | 103 |
| 4 | Carol | 104 |
| 5 | David | *null* |

5 rows

**Ergebnis:** Alle Kunden UND alle Bestellungen. Lücken werden mit NULL gefüllt.

**Use Case:** „Zeige alle Kunden UND alle Bestellungen, unabhängig von Beziehungen."

## 5.6 CROSS JOIN (Kartesisches Produkt)

**Regel:** Jede Zeile der ersten Tabelle wird mit jeder Zeile der zweiten kombiniert.

```sql
SELECT
  c.first_name,
  l.city
FROM customers c
CROSS JOIN locations l;
```

```
SELECT
  c.first_name,
  l.city
FROM customers c
CROSS JOIN locations l
```

| # | first_name | city |
|---|---|---|
| 1 | Alice | Berlin |
| 2 | Alice | Hamburg |
| 3 | Alice | Munich |
| 4 | Bob | Berlin |
| 5 | Bob | Hamburg |
| 6 | Bob | Munich |
| 7 | Carol | Berlin |
| 8 | Carol | Hamburg |
| 9 | Carol | Munich |
| 10 | David | Berlin |
| 11 | David | Hamburg |
| 12 | David | Munich |

12 rows

**Ergebnis:** 4 Kunden × 3 Städte = 12 Zeilen.

**Use Case:** Selten! Z.B. für Preis-Matrices oder kombinatorische Abfragen.

## 5.7 Self Join

**Regel:** Tabelle mit sich selbst joinen (z.B. Hierarchien).

```sql
-- Mitarbeiter-Hierarchie
CREATE TABLE employees (
  emp_id INTEGER PRIMARY KEY,
  name TEXT,
  manager_id INTEGER,
  FOREIGN KEY (manager_id) REFERENCES employees(emp_id)
);

INSERT INTO employees VALUES
```

```
  (1, 'CEO', NULL),
  (2, 'CTO', 1),
  (3, 'Dev Lead', 2),
  (4, 'Developer', 3);

-- Wer ist wessen Manager?
SELECT
  e.name AS employee,
  m.name AS manager
FROM employees e
LEFT JOIN employees m ON e.manager_id = m.emp_id;
```

```
-- Mitarbeiter-Hierarchie
CREATE TABLE employees (
  emp_id INTEGER PRIMARY KEY,
  name TEXT,
  manager_id INTEGER,
  FOREIGN KEY (manager_id) REFERENCES employees(emp_id)
)
```

*ok*

```
INSERT INTO employees VALUES
  (1, 'CEO', NULL),
  (2, 'CTO', 1),
  (3, 'Dev Lead', 2),
  (4, 'Developer', 3)
```

*ok*

```
-- Wer ist wessen Manager?
SELECT
  e.name AS employee,
  m.name AS manager
FROM employees e
LEFT JOIN employees m ON e.manager_id = m.emp_id
```

| # | employee | manager |
|---|---|---|
| 1 | CEO | *null* |
| 2 | CTO | CEO |
| 3 | Dev Lead | CTO |
| 4 | Developer | Dev Lead |

4 rows

## 5.8 Mehrere Joins kombinieren

```sql
-- Kunde → Bestellung → Positionen → Produkte
SELECT
  c.first_name,
  o.order_id,
  oi.quantity,
  p.product_name,
  p.price
FROM customers c
INNER JOIN orders o ON c.customer_id = o.customer_id
INNER JOIN order_items oi ON o.order_id = oi.order_id
INNER JOIN products p ON oi.product_id = p.product_id;
```

**Faustregel:** Jeder JOIN braucht eine ON-Bedingung!

## 5.9 Join-Cheat-Sheet

| Join-Typ | Ergebnis | Use Case |
|---|---|---|
| `INNER JOIN` | Nur übereinstimmende Zeilen | Kunden MIT Bestellungen |
| `LEFT JOIN` | Alle aus links + passende aus rechts | Alle Kunden (auch ohne Orders) |
| `RIGHT JOIN` | Alle aus rechts + passende aus links | Selten (wie LEFT, nur umgekehrt) |
| `FULL OUTER JOIN` | Alle aus beiden (Lücken = NULL) | Alle Kunden UND alle Orders |
| `CROSS JOIN` | Kartesisches Produkt (alle Kombinationen) | Kombinatorik, Matrizen |
| `SELF JOIN` | Tabelle mit sich selbst | Hierarchien, Vergleiche |

---

# 6. Set-Operationen

## 6.1 UNION – Ergebnisse vereinigen

```sql
-- Alle Kunden aus zwei Regionen
SELECT first_name, last_name FROM customers WHERE location_id = 1
UNION
```

```sql
SELECT first_name, last_name FROM customers WHERE location_id = 2;
```

```sql
-- Alle Kunden aus zwei Regionen
SELECT first_name, last_name FROM customers WHERE location_id = 1
UNION
SELECT first_name, last_name FROM customers WHERE location_id = 2
```

| # | first_name | last_name |
|---|---|---|
| 1 | Alice | Anderson |
| 2 | Bob | Brown |

2 rows

**UNION** entfernt Duplikate. **UNION ALL** behält sie.

## 6.2 INTERSECT – Schnittmenge

```sql
-- Kunden, die sowohl in Berlin wohnen als auch bestellt haben
SELECT customer_id FROM customers WHERE location_id = 1
INTERSECT
SELECT customer_id FROM orders;
```

```sql
-- Kunden, die sowohl in Berlin wohnen als auch bestellt haben
SELECT customer_id FROM customers WHERE location_id = 1
INTERSECT
SELECT customer_id FROM orders
```

| # | customer_id |
|---|---|
| 1 | 1 |

1 rows

## 6.3 EXCEPT – Differenz

```sql
-- Kunden OHNE Bestellungen
SELECT customer_id FROM customers
EXCEPT
SELECT customer_id FROM orders;
```

```
-- Kunden OHNE Bestellungen
SELECT customer_id FROM customers
EXCEPT
SELECT customer_id FROM orders
```

| # | customer_id |
|---|---|
| 1 | 4 |

1 rows

---

# 7. Best Practices

## 7.1 Query-Optimierung

✅ **Spalten explizit benennen** statt `SELECT *`

✅ **WHERE statt HAVING** (wenn möglich)

✅ **Aliase nutzen** für Lesbarkeit

✅ **Indexes auf Foreign Keys** (Performance)

✅ **LIMIT bei Exploration** (nicht alle Millionen Zeilen laden)

## 7.2 Schema-Design

✅ **Immer Primary Keys** definieren

✅ **Foreign Keys für Integrität** nutzen

✅ **NOT NULL für Pflichtfelder**

✅ **CHECK Constraints** für Validierung

✅ **DEFAULT-Werte** wo sinnvoll

✅ **Normalisierung bis 3NF** (außer Performance-Gründe)

## 7.3 Datenmanipulation

✅ **Immer WHERE bei UPDATE/DELETE** (außer wirklich alle Zeilen)

✅ **Transaktionen für Multi-Step-Operationen**

✅ **Bulk Insert statt einzelne INSERTs**

✅ **Backup vor riskanten Änderungen**

⚠️ **TRUNCATE nur wenn sicher** (meist nicht reversibel)

---

## 8. Häufige Fehler & Lösungen

### ❌ Fehler: UPDATE ohne WHERE

```
-- GEFAHR: Ändert ALLE Zeilen!
UPDATE customers SET name = 'Unknown';
```

✅ Immer WHERE nutzen:

```
UPDATE customers SET name = 'Unknown' WHERE customer_id = 5;
```

### ❌ Fehler: Kartesisches Produkt

```
-- GEFAHR: 1000 Kunden × 1000 Orders = 1 Million Zeilen!
SELECT * FROM customers, orders;
```

✅ Immer JOIN mit ON:

```
SELECT * FROM customers c
INNER JOIN orders o ON c.customer_id = o.customer_id;
```

### ❌ Fehler: NULL-Vergleich mit =

```
-- Funktioniert NICHT:
WHERE rating = NULL
```

✅ Nutze IS NULL:

```
WHERE rating IS NULL
```

### ❌ Fehler: Alias in WHERE

```
-- Fehler: price_with_tax nicht verfügbar!
SELECT price * 1.19 AS price_with_tax
FROM products
WHERE price_with_tax > 100;
```

✅ Berechnung in WHERE wiederholen:

```
SELECT price * 1.19 AS price_with_tax
FROM products
WHERE price * 1.19 > 100;
```

(Oder mit CTE/Subquery – siehe L10)

## 9. Weiterführende Themen

**Diese Themen kommen in späteren Sessions:**

- **Subqueries & CTEs** (L10+)
- **Window Functions** (L12)
- **Stored Procedures & Functions** (L13)
- **Transaktionen & ACID** (L11+)
- **Indexierung & Performance** (L15)
- **Views & Materialized Views** (L13)

## 10. Quick-Links zur Vorlesung

- **L7:** SQL Introduction & SELECT
- **L8:** DDL & DML
- **L9:** Normalisierung
- **L10:** Joins & Combining Data

## 📚 Zusätzliche Ressourcen

- SQL Standard (ISO/IEC 9075)
- PostgreSQL Documentation
- DuckDB Documentation
- SQLite Documentation
- DB-Fiddle (Online SQL Editor)

🎉 **Happy Querying!** Nutze dieses Cheat-Sheet als schnelle Referenz während der Übungen und Projekte!