# SQL: Übungsblatt

Selbst erstelltes Lernmaterial zum Lernvideo „SQL einfach erklärt: SELECT, WHERE, JOIN für die IHK-Prüfung“. Alle Namen, Orte und Beträge sind erfunden. Die Beispielausgaben gelten ohne ORDER BY mit nicht festgelegter Reihenfolge.

Name: _____ Datum: _____

## Die Tabellen des Pizzadienstes

**Kunde** (5 Zeilen)

| KundeNr | Name | Ort |
|---|---|---|
| 1 | Ada | Köln |
| 2 | Ben | Bonn |
| 3 | Cem | Köln |
| 4 | Dana | Essen |
| 5 | Emil | Köln |

**Bestellung** (6 Zeilen)

| BestellNr | KundeNr | Betrag |
|---|---|---|
| 101 | 1 | 18.50 |
| 102 | 3 | 32.00 |
| 103 | 1 | 24.00 |
| 104 | 2 | 12.90 |
| 105 | 4 | 41.00 |
| 106 | 3 | 9.50 |

**KundeNr** in der Tabelle Kunde: Primärschlüssel (jede Nummer nur einmal, nie leer). **KundeNr** in der Tabelle Bestellung: Fremdschlüssel, er zeigt auf den Kunden, der bestellt hat.

## Die sechs Regeln

1. **SELECT** wählt die Spalten, **FROM** nennt die Tabelle, **WHERE** filtert die Zeilen. Nenne nur die gefragten Spalten, `SELECT *` holt alle.
2. **Text** steht in **einfachen Anführungszeichen** (`'Köln'`), Zahlen nicht. Ohne sie hält die Datenbank Köln für eine Spalte und meldet einen Fehler.
3. Beim **JOIN** legt **ON** fest, was zusammengehört, zum Beispiel `ON k.KundeNr = b.KundeNr` (Primär- gleich Fremdschlüssel). Steht ein Name in beiden Tabellen, schreibst du `k.` oder `b.` davor. INNER JOIN: nur passende Paare. LEFT JOIN: alle Zeilen der linken Tabelle, fehlende Partner als NULL.
4. **Komma oder CROSS JOIN ohne Bedingung** liefern **jede Kombination** (Kreuzprodukt): 5 Kunden mal 6 Bestellungen sind 30 Zeilen. PostgreSQL und SQL Server lehnen ein JOIN ohne ON als Syntaxfehler ab.
5. **AND bindet stärker als OR**, ähnlich wie Punkt vor Strich: `A OR B AND C` heißt `A OR (B AND C)`. Setze Klammern: `(A OR B) AND C`.
6. **Fehlende Werte** prüfst du mit **IS NULL** oder IS NOT NULL, nie mit `= NULL`. NULL heißt fehlend oder unbekannt, nicht 0. Ein Vergleich mit NULL ergibt „unbekannt“, und WHERE lässt nur wahre Zeilen durch.

**Logische Reihenfolge der Verarbeitung:** 1. **FROM** mit JOIN … ON (Tabellen verbinden) → 2. **WHERE** (Zeilen filtern) → 3. **SELECT** (Spalten wählen) → 4. **ORDER BY** (sortieren). **Wer hat nie bestellt?** `LEFT JOIN … WHERE b.BestellNr IS NULL`

## Die Abfragen des Videos mit ihren Ergebnissen

| Abfrage | Zeilen | Bemerkung |
|---|---|---|
| `SELECT Name, Ort FROM Kunde` | 5 | |
| `SELECT Name FROM Kunde WHERE Ort = 'Köln'` | 3 | |
| `SELECT k.Name, b.Betrag FROM Kunde k JOIN Bestellung b ON k.KundeNr = b.KundeNr` | 6 | Emil fehlt: nie bestellt |
| `SELECT k.Name, b.Betrag FROM Kunde k, Bestellung b` | 30 | Kreuzprodukt: 5 × 6 |
| `SELECT k.Name, b.Betrag FROM Kunde k JOIN Bestellung b ON k.KundeNr = b.KundeNr WHERE k.Ort = 'Köln' AND b.Betrag > 20` | 2 | Cem 32,00 und Ada 24,00 |
| `SELECT k.Name, b.Betrag FROM Kunde k JOIN Bestellung b ON k.KundeNr = b.KundeNr WHERE k.Ort = 'Köln' OR k.Ort = 'Bonn' AND b.Betrag > 20` | 4 | Falle: ohne Klammern |
| `SELECT k.Name, b.Betrag FROM Kunde k JOIN Bestellung b ON k.KundeNr = b.KundeNr WHERE (k.Ort = 'Köln' OR k.Ort = 'Bonn') AND b.Betrag > 20` | 2 | mit Klammern |
| `SELECT k.Name, b.BestellNr FROM Kunde k LEFT JOIN Bestellung b ON k.KundeNr = b.KundeNr` | 7 | Emil mit NULL |
| `SELECT k.Name FROM Kunde k LEFT JOIN Bestellung b ON k.KundeNr = b.KundeNr WHERE b.BestellNr IS NULL` | 1 | nur Emil |
| `SELECT k.Name FROM Kunde k LEFT JOIN Bestellung b ON k.KundeNr = b.KundeNr WHERE b.BestellNr = NULL` | 0 | Falle: = NULL |

# Aufgaben A bis D

Schreibe die Abfrage in die Zeilen und trage das Ergebnis in die Tabelle ein. Die Tabellen sind die des Videos (Seite 1). Texte stehen in einfachen Anführungszeichen. Lösungen auf Seite 4.

### A. Zeige die BestellNr und den Betrag aller Bestellungen.

Deine Abfrage:

Ergebnis: 6 Zeilen

| BestellNr | Betrag |
|---|---|
| | |
| | |
| | |
| | |
| | |
| | |

### B. Welche Kunden wohnen nicht in Köln? Zeige Name und Ort.

Deine Abfrage:

Ergebnis: 2 Zeilen

| Name | Ort |
|---|---|
| | |
| | |

### C. Zeige BestellNr und Betrag aller Bestellungen unter 20 €, die größte zuerst. Die Reihenfolge zählt.

Deine Abfrage:

Ergebnis: 3 Zeilen

| BestellNr | Betrag |
|---|---|
| | |
| | |
| | |

### D. Zeige zu jeder Bestellung die BestellNr und den Namen des Kunden, sortiert nach der BestellNr. Die Reihenfolge zählt.

Deine Abfrage:

Ergebnis: 6 Zeilen

| BestellNr | Name |
|---|---|
| | |
| | |
| | |
| | |
| | |
| | |

## Richtig oder falsch?

| Aussage | richtig / falsch |
|---|---|
| AND bindet schwächer als OR. | |
| Die Komma-Schreibweise FROM Kunde k, Bestellung b ohne WHERE liefert 30 Zeilen. | |
| WHERE b.BestellNr = NULL findet die Kunden ohne Bestellung. | |
| NULL bedeutet dasselbe wie die Zahl 0. | |
| Text steht in SQL in einfachen Anführungszeichen. | |
| Der LEFT JOIN behält alle Zeilen der linken Tabelle. | |
| Ohne ORDER BY ist die Reihenfolge der Ergebniszeilen festgelegt. | |

# Aufgaben E bis G, Vorhersage und Fehlersuche

### E. Zeige Name und Betrag der Bestellungen von Kunden aus Köln, die unter 20 € liegen.

Deine Abfrage:

Ergebnis: 2 Zeilen

| Name | Betrag |
|---|---|
| | |
| | |

### F. Zeige Name und Betrag der Bestellungen von Kunden aus Köln oder Essen, die über 25 € liegen.

Deine Abfrage:

Ergebnis: 2 Zeilen

| Name | Betrag |
|---|---|
| | |
| | |

### G. Zeige alle Kunden aus Köln mit ihren BestellNr, auch die Kunden ohne Bestellung (Spalten: Name, BestellNr).

Deine Abfrage:

Ergebnis: 5 Zeilen

| Name | BestellNr |
|---|---|
| | |
| | |
| | |
| | |
| | |

## H. Wie viele Zeilen hat das Ergebnis?

Rechne im Kopf, ohne die Abfrage auszuführen. Denke an die Reihenfolge: erst FROM und JOIN, dann WHERE.

| Abfrage | Zeilen |
|---|---|
| 1. `SELECT * FROM Kunde k, Bestellung b WHERE k.Ort = 'Köln'` | |
| 2. `SELECT k.Name FROM Kunde k JOIN Bestellung b ON k.KundeNr = b.KundeNr WHERE k.Ort = 'Essen'` | |
| 3. `SELECT k.Name, b.BestellNr FROM Kunde k LEFT JOIN Bestellung b ON k.KundeNr = b.KundeNr WHERE k.Ort = 'Bonn'` | |
| 4. `SELECT k.Name FROM Kunde k LEFT JOIN Bestellung b ON k.KundeNr = b.KundeNr WHERE b.BestellNr = NULL` | |
| 5. `SELECT k.Name FROM Kunde k LEFT JOIN Bestellung b ON k.KundeNr = b.KundeNr` | |

## I. Fehler finden

Jede Abfrage soll etwas anderes liefern, als sie liefert. Nenne den Fehler und schreibe die Korrektur auf.

| Abfrage | Fehler und Korrektur |
|---|---|
| 1. `SELECT Name FROM Kunde WHERE Ort = Köln` | |
| 2. `SELECT k.Name, b.Betrag FROM Kunde k, Bestellung b WHERE k.Ort = 'Köln'` | |
| 3. `SELECT k.Name, b.Betrag FROM Kunde k JOIN Bestellung b ON k.KundeNr = b.KundeNr WHERE k.Ort = 'Köln' OR k.Ort = 'Essen' AND b.Betrag > 25` | |
| 4. `SELECT k.Name FROM Kunde k LEFT JOIN Bestellung b ON k.KundeNr = b.KundeNr WHERE b.BestellNr = NULL` | |
| 5. `SELECT KundeNr FROM Kunde k JOIN Bestellung b ON k.KundeNr = b.KundeNr` | |
| 6. `SELECT k.Name FROM Kunde k JOIN Bestellung b ON k.KundeNr = b.KundeNr WHERE b.BestellNr IS NULL` | |

# Lösungen

Alle Ergebnisse stammen aus demselben Rechenkern wie die interaktive Übungsseite und sind unabhängig nachgerechnet. Es gibt oft mehrere richtige Abfragen: Entscheidend ist das Ergebnis. Ohne ORDER BY ist die Reihenfolge der Zeilen nicht festgelegt.

### A (6 Zeilen)

```
SELECT BestellNr, Betrag FROM Bestellung
```

| BestellNr | Betrag |
|---|---|
| 101 | 18.50 |
| 102 | 32.00 |
| 103 | 24.00 |
| 104 | 12.90 |
| 105 | 41.00 |
| 106 | 9.50 |

### B (2 Zeilen)

```
SELECT Name, Ort FROM Kunde WHERE Ort <> 'Köln'
```

| Name | Ort |
|---|---|
| Ben | Bonn |
| Dana | Essen |

### C (3 Zeilen)

```
SELECT BestellNr, Betrag FROM Bestellung WHERE Betrag < 20 ORDER BY Betrag DESC
```

| BestellNr | Betrag |
|---|---|
| 101 | 18.50 |
| 104 | 12.90 |
| 106 | 9.50 |

### D (6 Zeilen)

```
SELECT b.BestellNr, k.Name FROM Kunde k JOIN Bestellung b ON k.KundeNr = b.KundeNr ORDER BY b.BestellNr
```

| BestellNr | Name |
|---|---|
| 101 | Ada |
| 102 | Cem |
| 103 | Ada |
| 104 | Ben |
| 105 | Dana |
| 106 | Cem |

### E (2 Zeilen)

```
SELECT k.Name, b.Betrag FROM Kunde k JOIN Bestellung b ON k.KundeNr = b.KundeNr WHERE k.Ort = 'Köln' AND b.Betrag < 20
```

| Name | Betrag |
|---|---|
| Ada | 18.50 |
| Cem | 9.50 |

### F (2 Zeilen)

```
SELECT k.Name, b.Betrag FROM Kunde k JOIN Bestellung b ON k.KundeNr = b.KundeNr WHERE (k.Ort = 'Köln' OR k.Ort = 'Essen') AND b.Betrag > 25
```

| Name | Betrag |
|---|---|
| Cem | 32.00 |
| Dana | 41.00 |

### G (5 Zeilen)

```
SELECT k.Name, b.BestellNr FROM Kunde k LEFT JOIN Bestellung b ON k.KundeNr = b.KundeNr WHERE k.Ort = 'Köln'
```

| Name | BestellNr |
|---|---|
| Ada | 101 |
| Ada | 103 |
| Cem | 102 |
| Cem | 106 |
| Emil | *NULL* |

## Lösungen H bis J

**H. Zeilen**

1. 18 Zeilen (3 Kunden aus Köln × 6 Bestellungen, Kreuzprodukt)
2. 1 Zeile
3. 1 Zeile
4. 0 Zeilen (= NULL ergibt nie wahr)
5. 7 Zeilen (Emil mit NULL)

**Richtig oder falsch**

1. falsch
2. richtig
3. falsch
4. falsch
5. richtig
6. richtig
7. falsch

**I. Fehler**

1. Text ohne einfache Anführungszeichen: Die Datenbank sucht eine Spalte Köln und meldet einen Fehler. Korrektur: `WHERE Ort = 'Köln'`
2. Kein ON: jede Zeile mit jeder, das Kreuzprodukt. Es kommen 18 Zeilen statt 4 Zeilen. Korrektur: `JOIN Bestellung b ON k.KundeNr = b.KundeNr statt Komma`
3. AND bindet stärker als OR: Köln mit jedem Betrag, oder Essen über 25. Es kommen 5 Zeilen statt 2 Zeilen. Klammern setzen. Korrektur: `WHERE (k.Ort = 'Köln' OR k.Ort = 'Essen') AND b.Betrag > 25`
4. Ein Vergleich mit = NULL ergibt nie „wahr“: 0 Zeilen. Richtig ist IS NULL. Korrektur: `WHERE b.BestellNr IS NULL`
5. Mehrdeutig: KundeNr steht in beiden Tabellen. Schreibe k.KundeNr oder b.KundeNr. Korrektur: `SELECT k.KundeNr`
6. Der INNER JOIN wirft Kunden ohne Bestellung schon beim Verbinden raus: 0 Zeilen. Es braucht einen LEFT JOIN. Korrektur: `LEFT JOIN statt JOIN`