Lernskript: SQL-Daten aggregieren mit `GROUP BY`

Nachdem man Daten filtern (`WHERE`) und verknüpfen (`JOIN`) kann, ist der nächste entscheidende Schritt, aus vielen einzelnen Datensätzen zusammenfassende Informationen zu generieren. Dies geschieht durch Aggregation.

Teil 1: Aggregatfunktionen

Aggregatfunktionen sind spezielle SQL-Funktionen, die eine Berechnung über eine Gruppe von Zeilen durchführen und einen einzigen Ergebniswert zurückliefern.

- **Die wichtigsten Funktionen:**
  - `COUNT(*)`: Zählt die **Anzahl der Zeilen** in einer Gruppe.
  - `SUM(spalte)`: Berechnet die **Summe** aller Werte in einer numerischen Spalte.
  - `AVG(spalte)`: Berechnet den **Durchschnittswert** aller Werte in einer numerischen Spalte.
  - `MAX(spalte)`: Findet den **höchsten Wert** in einer Spalte.
  - `MIN(spalte)`: Findet den **niedrigsten Wert** in einer Spalte.
- **Anwendung ohne `GROUP BY`:** Wenn man sie ohne `GROUP BY` verwendet, operieren sie auf der **gesamten Tabelle** und liefern genau eine Ergebniszeile.

```
-- Gibt den Gesamtumsatz aller Bestellungen zurück
SELECT SUM(PREIS) FROM BESTELLUNGEN;
```

Teil 2: Gruppierung mit GROUP BY

```
Konzept: Die GROUP BY-Klausel fasst Zeilen, die in einer bestimmten Spalte
denselben Wert haben, zu einer einzigen Gruppe zusammen. Die Aggregatfunktionen
operieren dann nicht mehr auf der ganzen Tabelle, sondern auf jeder dieser Gruppen
einzeln.

Analogie: Anstatt alle Schüler einer Schule zu zählen (COUNT(*)), gruppiert man
sie nach ihrer Klasse (GROUP BY KLASSE) und zählt dann die Schüler pro Klasse
(COUNT(*)).

Code-Beispiel:
code SQL

IGNORE_WHEN_COPYING_START
IGNORE_WHEN_COPYING_END
```

```
-- Berechnet die Anzahl der Bestellungen und den Umsatz FÜR JEDEN KUNDEN
SELECT
    KUNDE,
    COUNT(*) AS ANZAHL_BESTELLUNGEN,
    SUM(PREIS) AS UMSATZ_PRO_KUNDE
FROM
    BESTELLUNGEN
GROUP BY
    KUNDE;
```

Teil 3: Vertiefung (JavaMasta's Profi-Tipps)

```
Die goldene Regel von GROUP BY: Jede Spalte, die in der SELECT-Liste steht und
keine Aggregatfunktion ist (wie SUM, COUNT), MUSS in der GROUP BY-Klausel
aufgeführt sein. Ansonsten weiß die Datenbank nicht, welchen der gruppierten Werte
sie anzeigen soll.

WHERE vs. HAVING (entscheidender Unterschied):

    WHERE: Filtert einzelne Zeilen, BEVOR die Gruppierung und die Aggregation
stattfinden.
    code SQL
```

IGNORE_WHEN_COPYING_START IGNORE_WHEN_COPYING_END

-- Berücksichtige nur Bücher, BEVOR du den Umsatz pro Kunde berechnest SELECT KUNDE, SUM(PREIS) FROM BESTELLUNGEN WHERE KATEGORIE = 'Bücher' GROUP BY KUNDE;

HAVING: Filtert ganze Gruppen, NACHDEM die Aggregation stattgefunden hat. HAVING wird fast immer mit einer Aggregatfunktion verwendet. code SQL

```
IGNORE_WHEN_COPYING_START
IGNORE_WHEN_COPYING_END


-- Zeige nur die Kundengruppen an, deren Gesamtumsatz NACH der Berechnung über
100€ liegt
SELECT KUNDE, SUM(PREIS) FROM BESTELLUNGEN
GROUP BY KUNDE
HAVING SUM(PREIS) > 100;
```

Reihenfolge der Klauseln: Die logische Reihenfolge in einer SQL-Abfrage ist fast immer: SELECT ... FROM ... JOIN ... ON ... WHERE ... GROUP BY ... HAVING ... ORDER BY ....