

LU10: Aggregatfunktionen

Datensatz: Anzahl Einbrüche nach Gemeinden und Stadtkreisen des Kantons Zürich



Um die Code-Beispiele nachvollziehen zu können, brauchen Sie die Datenbank zh_einbrueche, die Sie hier als .zip-File downloaden können:

Zip-File mit Datenbank (sql-File) zu Einbrüchen im Kanton Zürich.

¹⁾

1. Aggregatfunktionen - Theorie & Motivation

Was sind Aggregatfunktionen?

Aggregatfunktionen berechnen **einen zusammengefassten Wert** über mehrere Zeilen (Records).
Typische Aufgaben:

- **Zählen** (Wie viele Einbrüche?)
- **Summieren** (Wie viele Fälle insgesamt?)
- **Durchschnitt** (Ø-Häufigkeitszahl pro Jahr)
- **Minimum/Maximum** (geringste/höchste Zahl pro Gemeinde)

Wozu braucht man das?

- **Datenanalyse & Reporting** (z.B. Lageberichte für Polizei/Medien)
- **Business Intelligence** (Trends, Hotspots je Gemeinde/Jahr)
- **Produkt-/Web-Integration** (APIs liefern Rohdaten → Aggregation im Backend/SQL spart App-Logik)
- **Entscheidungsgrundlagen** (Ressourcen planen, Präventionsmassnahmen priorisieren)

1.1 Wichtigste Aggregatfunktionen

Funktion	Zweck	NULLs
COUNT (*)	Zeilen zählen	zählt alle Zeilen
COUNT (spalte)	Nicht-NULL-Werte zählen	ignoriert NULL
SUM (spalte)	Summe	ignoriert NULL
AVG (spalte)	Durchschnitt	ignoriert NULL
MIN (spalte)	Minimum	ignoriert NULL
MAX (spalte)	Maximum	ignoriert NULL

Syntax

```
SELECT AGGREGATFUNKTION(ausdruck) AS alias
FROM zh_einbrueche.einbrueche;
```

1.2 Einstiegsbeispiele

Anzahl unterschiedlicher Gemeinden im Datensatz

```
SELECT COUNT(DISTINCT gemeindename) AS anzahl_gemeinden
FROM zh_einbrueche.einbrueche;
```

Ergebnis (Auszug)

anzahl_gemeinden
161

Höchster Einzelwert pro Zeile (Totalfälle)

```
SELECT MAX(straftaten_total) AS max_faelle_in_einer_zeile
FROM zh_einbrueche.einbrueche;
```

Ergebnis (Auszug)

max_faelle_in_einer_zeile
1060

2. GROUP BY - Daten zu Gruppen zusammenfassen

title	genre	qty		genre	total
book 1	adventure	4		adventure	7
book 2	fantasy	5		fantasy	8
book 3	romance	2		romance	3
book 4	adventure	3			
book 5	fantasy	3			
book 6	romance	1			

Warum ist GROUP BY wichtig? Ohne GROUP BY erhalten Sie **einen** Aggregatwert über **alle** Zeilen. Mit GROUP BY erhalten Sie **je Gruppe eine Zeile** (z.B. pro Jahr, pro Gemeinde oder pro Kombination aus Jahr+Gemeinde). Man kann so nach einzelnen Einträgen gruppieren.

Was darf in SELECT stehen?

Nur Aggregatfunktionen (z.B. SUM(...), AVG(...)) oder Spalten, die im GROUP BY aufgeführt sind.

Gültig:

```
SELECT ausgangsjahr, SUM(straftaten_total)
FROM zh_einbrueche.einbrueche
GROUP BY ausgangsjahr;
```

Ungültig (fehlendes GROUP BY für gemeindename):

```
SELECT gemeindename, SUM(straftaten_total)
FROM zh_einbrueche.einbrueche; -- führt zu Fehler/Zufallswerten
```

WHERE vs. HAVING

- **WHERE** filtert einzelne Zeilen vor dem Gruppieren.
- **HAVING** filtert fertige Gruppen nach dem Gruppieren.

Beispiel: Nur Jahre ab 2018 berücksichtigen (WHERE), und nur Gruppen mit Total > 1'000 zeigen (HAVING)

```
SELECT ausgangsjahr, SUM(straftaten_total) AS total_faelle
FROM zh_einbrueche.einbrueche
```

```
WHERE ausgangsjahr >= 2018 -- Zeilenfilter
GROUP BY ausgangsjahr
HAVING SUM(straftaten_total) > 15000; -- Gruppenfilter
```

Ergebnis: In den Jahren 2018, 2019 und 2024 haben über 15000 Einbrüche total stattgefunden.

Alias in GROUP BY

MySQL: Alias kann verwendet werden. Für Portabilität (z.B. andere SQL-Dialekte): Ausdruck wiederholen.

2.1 Beispiele: Gruppieren nach einem Kriterium

a) Total Einbrüche pro Jahr

```
SELECT ausgangsjahr,
SUM(straftaten_total) AS total_faelle
FROM zh_einbrueche.einbrueche
GROUP BY ausgangsjahr
ORDER BY ausgangsjahr;
```

Ergebnis (Auszug)

ausgangsjahr	total_faelle
2019	5080
2020	4525
2021	4789

b) Anzahl erfasste Gemeinden pro Jahr

```
SELECT ausgangsjahr,
COUNT(DISTINCT gemeindename) AS anzahl_gemeinden
FROM zh_einbrueche.einbrueche
GROUP BY ausgangsjahr
ORDER BY ausgangsjahr;
```

Ergebnis (Auszug)

ausgangsjahr	anzahl_gemeinden
2019	171
2020	171

ausgangsjahr	anzahl_gemeinden
2021	171

2.2 Beispiele: Gruppieren nach mehreren Kriterien

Total pro Jahr und Tatbestand (z.B. Einbruch, Einschleichen)

```
SELECT ausgangsjahr,  
tatbestand,  
SUM(straftaten_total) AS total_faelle  
FROM zh_einbrueche.einbrueche  
GROUP BY ausgangsjahr, tatbestand  
ORDER BY ausgangsjahr, tatbestand;
```

Ergebnis (Auszug)

ausgangsjahr	tatbestand	total_faelle
2020	Einbruch	3100
2020	Einschleichen	1425
2021	Einbruch	3250

3. HAVING - Gruppen gezielt filtern

WHERE kann **keine** Aggregatfunktionen enthalten. Wenn Sie **Ergebnisgruppen** (nach GROUP BY) filtern wollen, verwenden Sie **HAVING**.

Jahre mit über 5'000 Fällen insgesamt

```
SELECT ausgangsjahr,  
SUM(straftaten_total) AS total_faelle  
FROM zh_einbrueche.einbrueche  
GROUP BY ausgangsjahr  
HAVING SUM(straftaten_total) > 5000  
ORDER BY total_faelle DESC;
```

Ergebnis (Auszug)

ausgangsjahr	total_faelle
2013	6240
2012	6115

Abarbeitungsreihenfolge (vereinfacht) FROM → WHERE → GROUP BY → **Aggregatfunktionen** → HAVING → SELECT → ORDER BY.

4. Häufige Stolpersteine (und wie man sie vermeidet)

* SELECT enthält **ungegrupperte Spalten** → Fehlermeldung oder Zufallswerte.

→ Nur Spalten in GROUP BY ****oder**** in Aggregaten verwenden.

* **Division durch 0** bei Anteilen/Quoten.

→ NULLIF(denominator,0) verwenden.

* **MySQL-Spezialität:** Alias in GROUP BY ist erlaubt, aber **nicht portabel**.

→ Ausdruck wiederholen, wenn Portabilität wichtig ist.

* **NULLs:** COUNT(spalte) ignoriert NULL → **bewusste Spaltenwahl!**

¹⁾

Datenquelle: Kantonspolizei des Kantons Zürich

From:
<https://wiki.bzz.ch/> - BZZ - Modulwiki

Permanent link:
https://wiki.bzz.ch/modul/m290_guko/learningunits/lu10/theorie/a_einfuehrung?rev=1762379428

Last update: **2025/11/05 22:50**

