

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** – Rohdaten → verständliche Kennzahlen
- *Beispiel (Einbrüche ZH): ***Wie viele Fälle gab es pro Jahr?*** Nutzen: **Klare Jahreskurven für** Berichte, Dashboards **und** Kurzstatements (**z. B. für Polizei/Medien**). * Business Intelligence (BI) – **Trends & Hotspots erkennen** Beispiel (Einbrüche ZH): **Welche Gemeinden haben im Schnitt höhere Raten?** Nutzen: Prioritäten **setzen** (**Präventionskampagnen, Ressourceneinsatz, Fokus-Orte**). * Produkt-/Web-Integration – **APIs liefern Rohdaten, Apps brauchen Kennzahlen** Beispiel (YouTube Top 100): **Gesamt-Views je Channel → SUM(view_count) pro Channel; durchschnittliche Videolänge → AVG(duration) pro Channel.** Nutzen: **Das Backend liefert fertige Kennzahlen; die Frontend-Logik bleibt schlank und schnell.** * Entscheidungsgrundlagen – **faktenbasiert planen, vergleichen, priorisieren** Beispiel (Space Missions): **Welche Jahre sind startstark? → COUNT(*) pro Jahr; Erfolgsquote je Rakete → Anteil mission_status = 'Success' pro Rakete.** Nutzen: Vergleiche **über Jahre, Firmen, Raketen; bessere Budget-/Ressourcenentscheide.** ===== **1.1 Wichtigste Aggregatfunktionen** =====

Funktion	Zweck	NULLs
COUNT (*)	Zeilen zählen	zählt alle Zeilen

Funktion	Zweck	NULLs
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 gemeindenamen): `SELECT gemeindenamen, 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.

==== 2.1 Beispiele: Gruppieren nach einem Kriterium ====

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
2009	29972
2010	26770
2011	24394

... etc.

==== 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
2023	Einschleichdiebstahl	1961
2024	Einbruchdiebstahl	5956
2024	Einbrüche insgesamt	7998

... etc.

===== 3. HAVING - Gruppen filtern ===== WHERE kann keine **Aggregatfunktionen** enthalten. Wenn Sie Ergebnisgruppen (nach **GROUP BY**) filtern wollen, verwenden Sie HAVING.

Jahre mit über 25'000 Fällen insgesamt

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

```
GROUP BY ausgangsjahr  
HAVING SUM(straftaten_total) > 25000  
ORDER BY total_faelle DESC;
```

Ergebnis (Auszug)

ausgangsjahr	total_faelle
2009	29972
2012	28730
2010	26770

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=1762384598

Last update: 2025/11/06 00:16

