# SQL – Gruppierungen

## Was sind Gruppierungen?

Datensätze, die in einer Spalte oder mehreren Spalten die gleichen Werte aufweisen, können mithilfe des GROUP BY-Zusatzes in Gruppen zusammengefasst werden. Gruppierungen machen vor allem dann Sinn, wenn innerhalb der Gruppen mit den Spaltenwerten gerechnet wird, z.B. wenn die Gesamtzahl aller Artikel einer bestimmten Preiskategorie ermittelt werden soll.

| **Ohne Gruppierung** | **Mit Gruppierung** |
|---|---|
| `SELECT *`<br>`FROM artikel`<br>`ORDER BY APreis ASC` | `SELECT *`<br>`FROM artikel`<br>**`GROUP by APreis`**<br>`ORDER BY APreis ASC` |
| | |
| | `SELECT APreis,SUM(ABestand)`<br>`FROM artikel`<br>`GROUP BY APreis`<br>`ORDER BY APreis ASC` |
| | |

Alle Artikel mit dem Preis von 1,00 Euro werden durch die Gruppierung auf einen Datensatz „projiziert“. Im Falle der zweiten Gruppierungsklausel wird zusätzlich der Gesamtbestand aller Artikel einer Preiskategorie mithilfe der Aggregationsfunktion `SUM(.)` in der Spalte ausgegeben.

Genau genommen ist die Verwendung der Wildcard * zur Auswahl aller Spalten in Kombination mit GROUP BY nur bedingt sinnvoll, da diejenigen Spalten, nach denen nicht gruppiert wird, unterschiedliche Werte aufweisen können. Das Beispiel in Tabelle 1 zeigt, dass in einem solchen Fall die Werte eines (beliebigen) Datensatzes der Gruppe in diesen Spalten angezeigt wird. Bei restriktiverer Einstellung des Datenbanksystems müsste die Auswahl von Spalten, nach denen nicht gruppiert wird, eine Fehlermeldung liefern.

## Mehrere Gruppen

Man kann auch gleichzeitig nach mehreren Merkmalen gruppieren, indem man einen Ausdruck der Form

```
SELECT *  FROM verkaeufe GROUP BY jahr, land, produkt;
```

verwendet. Hier werden die Verkäufe nach Jahr, Land und Produkt gruppiert.

# Aliase für Tabellenspalten

Manchmal ist es praktisch, Tabellenspalten, Tabellen oder Berechnungen von Gruppierungen über einen Alias für das weitere SQL Statement nutzbar zu machen. Dafür dient das Schlüsselwort AS.

```
SELECT KundenID AS ID, KundenName AS Name
FROM Kunden WHERE Name LIKE '%er';
```

Damit entstehen die „temporären“ Tabellenspaltenaliase `ID` und `Name`, die im Statement verwendet werden können. Damit kann man die Länge von SQL Statements verkürzen oder den Abfragen mehr Bedeutung geben, indem man die Aliase geschickt benennt. Die Tabellenstruktur wird dadurch nicht verändert.

So kann man auch Berechnungen mit „sprechenden“ Aliasen versehen, was die Lesbarkeit erhöht:

```
SELECT AngestelltenID AS ID, AVG(AngestelltenGehalt) AS Durchschnittsgehalt
FROM Mitarbeiter WHERE Angestelltengehalt > 2000 GROUP BY Abteilung;
```

# Aufgaben

Löse die folgenden Aufgaben im SQL-Abfragefenster von phpMyAdmin auf der Datenbank webshop und speichere deine Lösungen in einer Textdatei oder deinem Info-Heft.

---

**(A1)**

Was erfragen die beiden SQL Abfragen im Abschnitt zum Thema Aliase? Welche Rückschlüsse lässt das auf die Struktur der Tabellen zu?

---

**(A2)**

Neben der Summenfunktion gibt es weitere Berechnungsfunktionen für zahlenwertige Spalten. Vervollständige die folgende Tabelle unter Verwendung folgender SQL-Abfrage:

```
  SELECT *,FUNKTION(ABestand)
  FROM artikel
  GROUP BY APreis
```

```
ORDER BY APreis
```

Dabei ist FUNKTION nacheinander durch AVG, COUNT, MAX, MIN und SUM zu ersetzen.

| Funktion | Bedeutung | Wert in Gruppe APreis=9.99 |
|---|---|---|
| AVG | | AVG(ABestand)= |
| COUNT | | COUNT(ABestand)= |
| MAX | | MAX(ABestand)= |
| MIN | | MIN(ABestand)= |
| SUM | | SUM(ABestand)= |

---

**(A3)**

Importiere die Tabellen der Datenbank

webshop.zip

in deine Datenbank, um die folgenden Aufgaben zu lösen. Verwende, wo nötig Aliase.

(i) Gib den jeweiligen Gesamtbestand der Artikel in den verschiedenen Preiskategorien unter 10,00EUR an.

(ii) Gib Gruppen mit gleichem Bestand und Preis zurück.

(iii) Notiere ohne phpMyAdmin zu verwenden, was die folgende SQL-Abfrage ausgibt:

```
SELECT ABestand,COUNT(*)
FROM artikel
GROUP BY ABestand
ORDER BY ABestand
```

(iv) Informiere dich über die HAVING-Bedingung und löse damit Teilaufgabe (i) erneut. Erläutere den Unterschied zwischen WHERE und HAVING.

(v) Gib alle Preiskategorien aus, in denen der maximal erzielbare Umsatz über 3.000,00 Euro liegt. Ist die Aufgabe auch mittels WHERE anstelle von HAVING lösbar?

(Lösungen)

---

**(A4)**

Arbeite nun wieder mit unserer Adressdatenbank, verwende, wo nötig Aliase.

1. Gib die Zahl der Personen an, die in den verschiedenen Kundenkategorien sind.
2. Welche durchschnittliche Bonuspunktzahl haben die Mitglieder der verschiedenen Kunden-Kategorien?
3. Gruppiere die Adressliste nach Kunden-Kategorien und Bonuspunkte (In einer Gruppe sollen also alle Datensätze sein, die dieselbe Kundenkategorie und dieselbe Bonunspunktzahl haben). Gibt auf diese Weise die Zahl der Personen in der jeweiligen Gruoppe, die Kategorie und die Zahl der Bonuspunkte aus.
4. Gib bei der vorherigen Abfrage nur die Gruppen aus, die mehr als ein Mitglied haben. Geht das mit einem `WHERE` Statement?
5. Erzeuge eine Liste mit der Zahl der Gold-Kunden mit mehr als 1000 Bonuspunkten gruppiert nach Stadt.

From:
https://wiki.qg-moessingen.de/ - **QG Wiki**

Permanent link:
**https://wiki.qg-moessingen.de/faecher:informatik:oberstufe:datenbanken:sql_gruppierungen:start?rev=1604576237**

Last update: **05.11.2020 12:37**