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

```
SELECT *
FROM artikel
ORDER BY APreis ASC
```

	ANr	AName	APreis	ABild	ABestand
Bearbeiten Kopieren Löschen	100009	Visual C# 2010 Shortcuts - Befehlskarte	1.00	93970143A.jpg	100
Bearbeiten Kopieren Löschen	100011	TYPO3 Referenz: Extbase & Fluid - Befehlskarte	1.00	00000233A.jpg	100
Bearbeiten Kopieren Löschen	100010	Visual Basic 2010 Shortcuts - Befehlskarte	1.00	93970145A.jpg	100
Bearbeiten Kopieren Löschen	100008	CSS Referenz - Befehlskarte	1.00	93970156A.jpg	100
Bearbeiten Kopieren Löschen	100007	HTML Referenz - Befehlskarte	1.00	93970155A.jpg	100
Bearbeiten Kopieren Löschen	100006	XML & DTD Referenz - Befehlskarte	1.00	93970135A.jpg	100
Bearbeiten Kopieren Löschen	100005	C# .NET Referenz - Befehlskarte	1.00	93970130A.jpg	100
Bearbeiten Kopieren Löschen	100004	Java Referenz - Befehlskarte	1.00	93970138A.jpg	100
Bearbeiten Kopieren Löschen	100003	XML Schema Referenz - Befehlskarte	1.00	93970128A.jpg	70
Bearbeiten Kopieren Löschen	100002	PHP Referenz - Befehlskarte	1.00	93970129A.jpg	0
Bearbeiten Kopieren Löschen	100012	VBA mit Excel - easy	2.99	00000233A.jpg	50
Bearbeiten Kopieren Löschen	100016	Programmieren lernen für Kinder	9.99	82724415.jpg	80
Bearbeiten Kopieren Löschen	100021	Datenbanken: Implementierungstechniken	9.99	82661438.jpg	70
Bearbeiten Kopieren Löschen	100023	MySQL 5 - kurz & gut	9.99	89721525A.jpg	100
Bearbeiten Kopieren Löschen	100020	MySQL - Das große Buch	9.99	23679192BT.jpg	70
Bearbeiten Kopieren Löschen	100001	Softwareentwicklung - Einstieg für Anspruchsvolle	14.99	82732851.jpg	1
Bearbeiten Kopieren Löschen	100024	PHP 5.3 & MySQL 5.1	19.99	21293244T.jpg	34
Bearbeiten Kopieren Löschen	100022	Datenbanken - Grundlagen und Design	19.99	82665529.jpg	60

Mit Gruppierung

```
SELECT *
FROM artikel
GROUP by APreis
ORDER BY APreis ASC
```

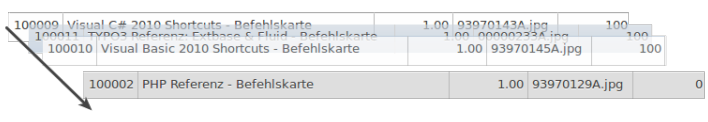
	ANr	AName	APreis	ABild	ABestand
Bearbeiten Kopieren Löschen	100002	PHP Referenz - Befehlskarte	1.00	93970129A.jpg	0
Bearbeiten Kopieren Löschen	100012	VBA mit Excel - easy	2.99	00000233A.jpg	50
Bearbeiten Kopieren Löschen	100016	Programmieren lernen für Kinder	9.99	82724415.jpg	80
Bearbeiten Kopieren Löschen	100001	Softwareentwicklung - Einstieg für Anspruchsvolle	14.99	82732851.jpg	1
Bearbeiten Kopieren Löschen	100013	CSS - Basis-Know-how	19.99	23351670BT.jpg	40
Bearbeiten Kopieren Löschen	100000	Handbuch der Java-Programmierung	29.99	82732874.jpg	20
Bearbeiten Kopieren Löschen	100019	SOA - Entwurfsprinzipien für serviceorientierte Ar...	39.99	24918456BT.jpg	80

```
SELECT APreis,SUM(ABestand)
FROM artikel
GROUP BY APreis
ORDER BY APreis ASC
```

APreis	SUM(ABestand)
1.00	870
2.99	50
9.99	320
14.99	1
19.99	686
29.99	197
39.99	80

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.

100009	Visual C# 2010 Shortcuts - Befehlskarte	1.00	93970143A.jpg	100
100011	TYPO3 Referenz: Extbase & Fluid - Befehlskarte	1.00	00000233A.jpg	100
100010	Visual Basic 2010 Shortcuts - Befehlskarte	1.00	93970145A.jpg	100
100008	CSS Referenz - Befehlskarte	1.00	93970156A.jpg	100
100007	HTML Referenz - Befehlskarte	1.00	93970155A.jpg	100
100006	XML & DTD Referenz - Befehlskarte	1.00	93970135A.jpg	100
100005	C# .NET Referenz - Befehlskarte	1.00	93970130A.jpg	100
100004	Java Referenz - Befehlskarte	1.00	93970138A.jpg	100
100003	XML Schema Referenz - Befehlskarte	1.00	93970128A.jpg	70
100002	PHP Referenz - Befehlskarte	1.00	93970129A.jpg	0
100012	VBA mit Excel - easy	2.99	00000233A.jpg	50
100016	Programmieren lernen für Kinder	9.99	82724415.jpg	80
100021	Datenbanken: Implementierungstechniken	9.99	82661438.jpg	70
100023	MySQL 5 - kurz & gut	9.99	89721525A.jpg	100
100020	MySQL - Das große Buch	9.99	23679192BT.jpg	70
100001	Softwareentwicklung - Einstieg für	14.99	82732851.jpg	1



Alle Artikel mit dem Preis 1.00 werden "übereinander" gelegt. Sichtbar ist nur "die oberste Schicht", ein zufälliger Datensatz (in diesem Beispiel der letzte).

Während der Gruppierung kann kann aber z.B. mit SUM(ABestand) der Gesamtbestand einer Gruppe bestimmt werden. Oder mit AVG der Durchschnittspreis...

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 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?

Importiere die Tabellen der Datenbank

webshop.sql.zip

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

(A2)

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

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

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

- Notiere die allgemeine Bedeutung der Aggregationsfunktion.
- Halte die Ausgabe für die Preisgruppe `APreis=9.99` fest.
- Überlege dir jeweils, was die Ausgabe bedeutet und formuliere einen Satz der, die Bedeutung für die Preisgruppe `APreis=9.99` beschreibt.

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

(A3)

(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 (Gold, Silber, Bronze oder kein Status) sind.
   Lösung `SELECT Kundenstatus, COUNT(`Kundenstatus`) FROM `adressen` GROUP BY `Kundenstatus`;`
2. Welche durchschnittliche Bonuspunktzahl haben die Mitglieder der verschiedenen Kunden-Kategorien?
   Lösung `SELECT Kundenstatus, COUNT(`Kundenstatus`), AVG(`Bonuspunkte`) AS Durchschnittspunkte FROM `adressen` GROUP BY `Kundenstatus`;`
3. Gruppiere die Adressliste nach Kunden-Kategorien und Bonuspunkte (In einer Gruppe sollen also alle Datensätze sein, die dieselbe Kundenkategorie und dieselbe Bonuspunktzahl haben). Gibt auf diese Weise die Zahl der Personen in der jeweiligen Gruppe, die Kategorie und die Zahl der Bonuspunkte aus.
   Lösung `SELECT`Kundenstatus`, `Bonuspunkte`, COUNT(*) AS Kundenzahl FROM `adressen` GROUP BY `Kundenstatus`, `Bonuspunkte`;`
4. Gib bei der vorherigen Abfrage nur die Gruppen aus, die mehr als ein Mitglied haben. Geht das mit einem WHERE Statement?
   Lösung `SELECT`Kundenstatus`, `Bonuspunkte`, COUNT(*) AS Kundenzahl FROM `adressen` GROUP BY `Kundenstatus`, `Bonuspunkte` HAVING Kundenzahl > 1;`
   Das geht nicht mit WHERE, da die Kundenzahl in den Gruppen erst nach der Gruppierung bekannt ist.
5. Erzeuge eine Liste mit der Zahl der Gold-Kunden mit mehr als 1000 Bonuspunkten gruppiert nach Stadt, absteigend sortiert nach der Zahl der Bonuspunkte.
   Lösung `SELECT `Stadt`, `Kundenstatus`, `Bonuspunkte`, COUNT(*) AS`

```
Kundenzahl FROM `adressen` WHERE `Kundenstatus` = "Gold" AND
`Bonuspunkte` > 1000 GROUP BY `Stadt` ORDER BY `Bonuspunkte` DESC;
```

6. Erzeuge eine Liste mit der Zahl aller Kunden, die keine leere Kundenkategorie haben mit mehr als 1000 Bonuspunkten gruppiert nach Stadt und Kategorie, absteigend sortiert nach der Zahl der gesamten Bonuspunkte für die jeweilige Gruppe.
   Lösung `SELECT `Stadt`, `Kundenstatus`, `Bonuspunkte`, COUNT(*) AS Kundenzahl, SUM(`Bonuspunkte`) AS Punktesumme FROM `adressen` WHERE `Kundenstatus` != "" AND `Bonuspunkte`> 1000 GROUP BY `Stadt`, `Kundenstatus` ORDER BY Punktesumme DESC;`

Material

02-sql-select.pdf	744.2 KiB	10.11.2020 07:37
auswahl_001.png	143.4 KiB	14.11.2019 13:00
auswahl_005.png	9.2 KiB	14.11.2019 13:00
auswahl_006.png	58.1 KiB	14.11.2019 13:00
erklaerung.png	133.5 KiB	14.11.2019 13:00

From:
https://info-bw.de/ -

Permanent link:
https://info-bw.de/faecher:informatik:oberstufe:datenbanken:sql_gruppierungen:start

Last update: **18.01.2024 08:57**