Die „erweiterte“ SELECT-Anweisung

Eine SELECT-Anweisung besteht aus folgenden Klauseln:

SELECT **<**ALL | DISTINCT**>** **<**Feldliste | * | Aggregatfunktion**>**
FROM <Tabellenliste>
[WHERE <Abfragebedingung>]
[GROUP BY <GruppierungsFeldListe>]
[HAVING <Gruppenauswahlkriterium>]
[ORDER BY Feld1 [ASC] | [DESC] [, Feld2 [ASC] | [DESC] …]]

Klauseln in eckigen Klammern sind optional und müssen nicht angegeben werden. Doch erklären wir das lieber an einem Beispiel.

1. Ein fürchterlicher Sturm hat die Autowerkstatt verwüstet. Sie sollen eine Liste mit dem Schaden je Lagerort erstellen. Gehen Sie davon aus, das alle Artikel im Bestand unbrauchbar sind.

`Versicherungsschaden:`

```
SELECT Lagerort, SUM( Preis * Bestand )  AS Schaden
FROM Bestand
WHERE ArtNr != 999999
GROUP  BY Lagerort

SELECT Lagerort, ROUND(SUM( Preis * Bestand ),3)  AS
Schaden
FROM Bestand
WHERE ArtNr != 999999
GROUP  BY Lagerort
```

Ergebnis:

Lagerort	Schaden
100	1054.360
110	1496.620
120	2350.580
130	5494.030
140	2449.450
454	1612.700

Es wird nach dem Feld '`Lagerort`' gruppiert. Für alle Datensätze mit gleichem Lagerort wird mit Hilfe der Aggregatfunktion 'SUM( )' der Versicherungsschaden ermittelt.

Aggregatfunktionen:

COUNT()	-	Anzahl
SUM()	-	Summe
MIN()	-	Minimalwert
MAX()	-	Maximalwert
AVG()	-	Mittelwert

Aggregatfunktionen führen Berechnungen für eine Wertemenge durch und geben einen einzelnen Wert zurück.
Durch weitere Funktionen kann das Ergebnis ansprechend formatiert werden:

ROUND(Spalte, Anzahl Kommastellen)
CONCAT(ZeichenKette1, ZeichenKette2, …)

```
SELECT Lagerort,
       CONCAT(ROUND(SUM( Preis * Bestand),3), " €") AS Schaden
FROM Bestand
WHERE ArtNr <> 999999
GROUP  BY Lagerort
```

Lagerort	Schaden
100	1054.360 €
110	1496.620 €
120	2350.580 €
130	5494.030 €
140	2449.450 €
454	1612.700 €

2. Inventur:
Die Auszubildenden dieser Werkstatt sollen eine Inventur durchführen. Je zwei Azubis zählen die Artikel eines Lagerortes. Sind sie mit der Inventur fertig, können sie nach Hause gehen. Welchen Lagerort würden Sie sich aussuchen?
Hinweis: Kleinteile werden gewogen.

Lagerort	Anzahl_Artikel
100	10
130	5
120	4
140	3
454	3
110	2
	1

```
SELECT Lagerort, COUNT(*) AS Anzahl_Artikel
FROM Bestand
GROUP BY Lagerort
ORDER BY Anzahl_Artikel DESC
```

Die letzte Zeile entsteht durch die Arbeitszeit, die keinem Lagerort zugeordnet ist. Schließen Sie diese Zeile noch vom Ergebnis aus.

3. Sommerfest:
Der Besitzer der Werkstatt plant ein Sommerfest für seine Kunden, um etwas Werbung für seinen Betrieb zu machen. Dazu möchte er von Ihnen ein Liste mit der Anzahl der Kunden je Wohnort, damit er Stühle und Tische bestellen kann. Die Liste soll absteigend nach der Anzahl sortiert sein.

Wohnort	Anzahl_Kunden
Leipzig	30
Wurzen	6
Grimma	4
Beucha	3
Taucha	3
Eilenburg	3
Schkeuditz	3

Oh Gott, die Kunden aus Grimma und Beucha kloppen sich immer. Die sollen Sie von der Liste ausschließen.

Oh je, das wird teuer. Dazu ist der Werkstattinhaber viel zu geizig. Eigentlich haben wir in Leipzig schon genug Kunden. Schließen wir mal alle Wohnorte mit mehr als fünf Kunden von der Liste aus.

<u>HAVING – Klausel:</u>

Die letzte Aufgabe ist ohne eine weitere Klausel nicht lösbar. Mit der `WHERE`-Klausel kann ich nur Datensätze aufgrund von Feldinhalten filtern. z.B.

```
...
WHERE NOT (Wohnort = 'Beucha' OR Wohnort='Grimma')
...
```

Um alle Kunden aus diesen beiden Städten auszuschließen.

Wohnort	Anzahl_Kunden
Schkeuditz	3
Eilenburg	3
Taucha	3
Markkleeberg	2
Kühren	2
Wedelwitz	2
Panitzsch	2

```
SELECT Wohnort, COUNT(*) AS Anzahl_Kunden
FROM Kunden
WHERE NOT (Wohnort = 'Beucha' OR Wohnort='Grimma')
GROUP BY Wohnort
HAVING Anzahl_Kunden <= 5
ORDER BY Anzahl_Kunden DESC
```

Die `HAVING`-Klausel wird im Zusammenhang mit Aggregatfunktionen benutzt, um ein zuvor berechnetes Ergebnis zu filtern. In unserem Fall werden alle Zeilen, in denen die Aggregatfunktion `COUNT()` eine Anzahl ermittelt, die nicht die Bedingung erfüllt, ausgeschlossen.

4. Ermitteln Sie die Umsätze je Monat und Jahr. Sortieren Sie die Ausgabe nach Jahr absteigend und nach Monat aufsteigend.

```
SELECT YEAR( AuftrDat ) AS Jahr,
       MONTH( AuftrDat ) AS Monat,
       ROUND(SUM( RechBetrag ),2) AS Umsatz
FROM auftrag
GROUP BY  Jahr, Monat
ORDER BY Jahr DESC, Monat
```

Jahr	Monat	Umsatz
2000	1	0.00
2000	8	0.00
2000	9	0.00
2000	10	0.00
2000	11	0.00
1999	2	1000.00
1999	3	20.99

Wie sie sehen, kann in der `GROUP BY` – Klausel auch eine Liste von Feldnamen stehen.