Datenbanken und SQL

(Woche 3 - Tag 3)

Agenda

- Gruppierung
  - Definition + Motivation
  - Beispiele
- HAVING-Klausel
  - Definition + Motivation
  - Beispiele

Gruppierung

Definition + Motivation

- Wir haben bisher Aggregat-Funktionen genutzt, um **einzelne Werte** zu ermitteln.
- Entsprechend waren daher die Ausgaben auch jeweils nur **1-zeilig**.
- Dies lag daran, dass wir bei den bisherigen Abfragen alle betrachteten Datensätze – anschaulich gesprochen – **„in einen Topf warfen“** und von diesen dann „die“ Summe (oder „den“ Durchschnittswert, „das“ Minimum … etc.) berechnen ließen.
- Im Folgenden wollen wir uns hierzu eine Variante anschauen, bei der wir die zu berücksichtigenden Datensätze „gruppieren, um daraufhin die zu ermittelnden Aggregatwerte **pro Gruppe** berechnen zu lassen.
- Auf diese Weise werden wir die ermittelten Ergebnisse pro Gruppe vergleichen (oder gegebenenfalls auch sortieren) können und jedenfalls in die Lage versetzt, deutlich **interessantere Abfragen** zu formulieren.

Beispiele

Vorname, Nachname und Anzahl der Abrechnungen **PRO Kunde** **(repräsentiert durch Vor- und Nachname)**:

```
SELECT Vorname, Nachname, COUNT(*) FROM Kunde, Abrechnung
WHERE Kunde.Kunde_ID=Abrechnung.Kunde_ID
GROUP BY Vorname, Nachname;
```

Vorname	Nachname	COUNT(*)
Elli	Rot	2
Eva	Hahn	1
Peter	Kaufnix	1
Rita	Myrnow	1
Vera	Deise	2
Witali	Myrnow	3

Vorname, Nachname und Anzahl der Abrechnungen **PRO Kunde** **(repräsentiert durch Vor- und Nachname)**:
(**sortiert** nach Anzahl der Abrechnungen **absteigend**)

```
SELECT Vorname, Nachname, COUNT(*) FROM Kunde, Abrechnung
WHERE Kunde.Kunde_ID=Abrechnung.Kunde_ID
GROUP BY Vorname, Nachname
ORDER BY COUNT(*) DESC;
```

Vorname	Nachname	COUNT(*)
Witali	Myrnow	3
Elli	Rot	2
Vera	Deise	2
Rita	Myrnow	1
Eva	Hahn	1
Peter	Kaufnix	1

Die **Repräsentation eines Kunden** mittels Vor- und Nachnamen ist nicht unproblematisch, da es unterschiedliche, aber gleichnamige Kunden geben könnte, deren Abrechnungen dann fälschlicherweise **„in einen (gemeinsamen) Topf“** geworfen werden würden.

Beispiele

Nachname und Anzahl der Abrechnungen **PRO Kunde** **(repräsentiert durch Vor- und Nachname)**:

```
SELECT Vorname, Nachname, COUNT(*) FROM Kunde, Abrechnung
WHERE Kunde.Kunde_ID=Abrechnung.Kunde_ID
GROUP BY Vorname, Nachname;
```

Vorname	Nachname	COUNT(*)
Elli	Rot	2
Eva	Hahn	1
Peter	Kaufnix	1
Rita	Myrnow	1
Vera	Deise	2
Witali	Myrnow	3

Nachname und Anzahl der Abrechnungen **PRO Kunde** **(repräsentiert durch Vor- und Nachname)**:
(**sortiert** nach Anzahl der Abrechnungen **absteigend**)

```
SELECT Vorname, Nachname, COUNT(*) FROM Kunde, Abrechnung
WHERE Kunde.Kunde_ID=Abrechnung.Kunde_ID
GROUP BY Vorname, Nachname
ORDER BY COUNT(*) DESC;
```

Vorname	Nachname	COUNT(*)
Witali	Myrnow	3
Elli	Rot	2
Vera	Deise	2
Rita	Myrnow	1
Eva	Hahn	1
Peter	Kaufnix	1

Die **Repräsentation eines Kunden** mittels Vor- und Nachnamen ist nicht unproblematisch, da es unterschiedliche, aber gleichnamige Kunden geben könnte, deren Abrechnungen dann fälschlicherweise **„in einen (gemeinsamen) Topf“** geworfen werden würden.

Im obigen Fall wäre es also günstiger:

a) die Kunden-ID ausgeben zu lassen
b) nach der Kunden-ID zu gruppieren

Beispiele

Nachname und Anzahl der Abrechnungen **PRO Kunde** **(repräsentiert durch Vor- und Nachname)**:

```
SELECT Vorname, Nachname, COUNT(*) FROM Kunde, Abrechnung
WHERE Kunde.Kunde_ID=Abrechnung.Kunde_ID
GROUP BY Vorname, Nachname;
```

Vorname	Nachname	COUNT(*)
Elli	Rot	2
Eva	Hahn	1
Peter	Kaufnix	1
Rita	Myrnow	1
Vera	Deise	2
Witali	Myrnow	3

Nachname und Anzahl der Abrechnungen **PRO Kunde** **(repräsentiert durch Vor- und Nachname)**:
(**sortiert** nach Anzahl der Abrechnungen **absteigend**)

```
SELECT Vorname, Nachname, COUNT(*) FROM Kunde, Abrechnung
WHERE Kunde.Kunde_ID=Abrechnung.Kunde_ID
GROUP BY Vorname, Nachname
ORDER BY COUNT(*) DESC;
```

Vorname	Nachname	COUNT(*)
Witali	Myrnow	3
Elli	Rot	2
Vera	Deise	2
Rita	Myrnow	1
Eva	Hahn	1
Peter	Kaufnix	1

Die **Repräsentation eines Kunden** mittels Vor- und Nachnamen ist nicht unproblematisch, da es unterschiedliche, aber gleichnamige Kunden geben könnte, deren Abrechnungen dann fälschlicherweise **„in einen (gemeinsamen) Topf“** geworfen werden würden.

Im obigen Fall wäre es also günstiger:

a) die Kunden-ID ausgeben zu lassen
b) nach der Kunden-ID zu gruppieren

Dies ist aber dann bereits Teil der Aufgabenstellung und muss bzgl. einer IHK-Aufgabe nicht berücksichtigt werden. So kann für die IHK die folgende Empfehlung ausgesprochen werden: Notieren Sie **hinter GROUP BY** einfach alle Attribute, die auch **hinter SELECT** notiert wurden, **MIT AUSNAHME** der Attribute aller aufgeführten Aggregatfunktionen.

HAVING

Definition + Motivation

- Wie schon WHERE und ON ist auch **HAVING** eine „Bedingungs-Klausel“.
- Sie ist **obligatorisch**, wenn die Bedingung von einem **aggregierten Wert** spricht.
- Auch diese Funktionalität wird uns erlauben, **interessantere Abfragen** zu formulieren. Dies werden wir im Folgenden durch einige Beispiele illustrieren.

Beispiele

PRO Hersteller: Hersteller-Name und Preis seines teuersten Produkts (im Sortiment von „Geld_her“).
(Es sollen aber nur Hersteller berücksichtigt werden, deren **teuerstes Produkt mehr als 30 Euro** kostet.)

```
SELECT Hersteller_Name, MAX(Euro_Preis) FROM Produkt, Hersteller
WHERE Produkt.Hersteller_ID=Hersteller.Hersteller_ID
GROUP BY Hersteller_Name
HAVING MAX(Euro_Preis) > 30;
```

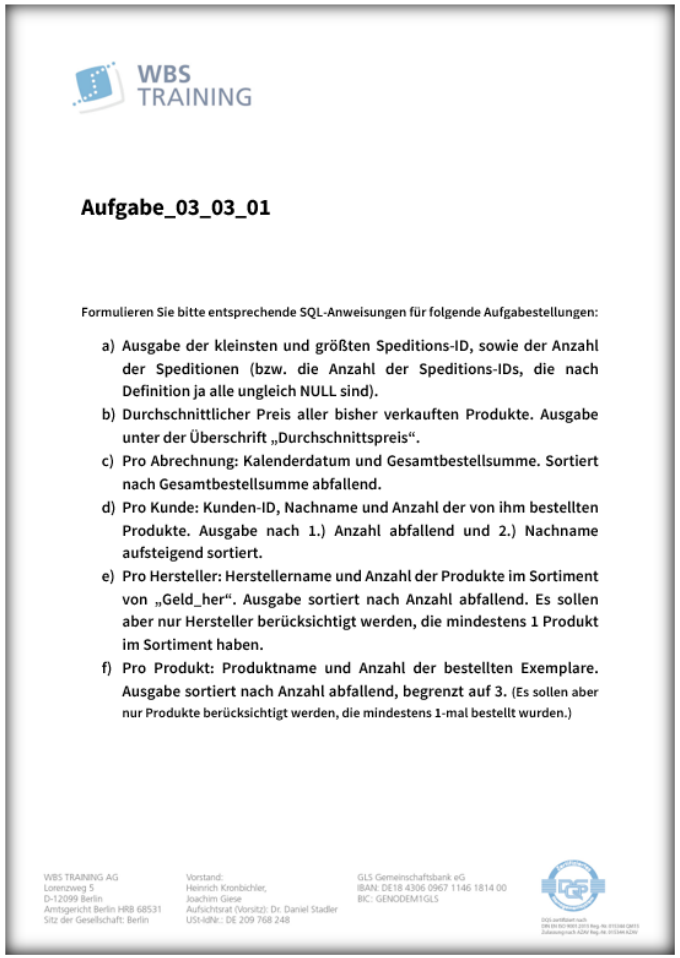
Hersteller_Name	MAX(Euro_Preis)
Contrabit	45.05
Ladenhut AG	1000.00
UltraBug	98.00

PRO Hersteller: Hersteller-Name und Preis seines teuersten Produkts (im Sortiment von „Geld_her“).
(Es sollen aber nur Hersteller berücksichtigt werden, deren **Produkte im Durchschnitt weniger als 500 Euro** kosten.)

```
SELECT Hersteller_Name, MAX(Euro_Preis) FROM Produkt, Hersteller
WHERE Produkt.Hersteller_ID=Hersteller.Hersteller_ID
GROUP BY Hersteller_Name
HAVING AVG(Euro_Preis) < 500;
```

Hersteller_Name	MAX(Euro_Preis)
AntiByte	22.75
Contrabit	45.05
UltraBug	98.00

Gemeinsame Übung („Live-Coding“) -> A_03_03_01



WBS TRAINING

Aufgabe_03_03_01

Formulieren Sie bitte entsprechende SQL-Anweisungen für folgende Aufgabestellungen:

a) Ausgabe der kleinsten und größten Speditions-ID, sowie der Anzahl der Speditionen (bzw. die Anzahl der Speditions-IDs, die nach Definition ja alle ungleich NULL sind).
b) Durchschnittlicher Preis aller bisher verkauften Produkte. Ausgabe unter der Überschrift „Durchschnittspreis“.
c) Pro Abrechnung: Kalenderdatum und Gesamtbestellsumme. Sortiert nach Gesamtbestellsumme abfallend.
d) Pro Kunde: Kunden-ID, Nachname und Anzahl der von ihm bestellten Produkte. Ausgabe nach 1.) Anzahl abfallend und 2.) Nachname aufsteigend sortiert.
e) Pro Hersteller: Herstellername und Anzahl der Produkte im Sortiment von „Geld_her“. Ausgabe sortiert nach Anzahl abfallend. Es sollen aber nur Hersteller berücksichtigt werden, die mindestens 1 Produkt im Sortiment haben.
f) Pro Produkt: Produktname und Anzahl der bestellten Exemplare. Ausgabe sortiert nach Anzahl abfallend, begrenzt auf 3. (Es sollen aber nur Produkte berücksichtigt werden, die mindestens 1-mal bestellt wurden.)

WBS TRAINING AG
Lorenzweg 5
D-12099 Berlin
Amtsgericht Berlin HRB 68531
Sitz der Gesellschaft: Berlin

Vorstand:
Heinrich Kronbichler,
Joachim Giese
Aufsichtsrat (Vorsitz): Dr. Daniel Stadler
USt-IdNr.: DE 209 768 248

GLS Gemeinschaftsbank eG
IBAN: DE18 4306 0967 1146 1814 00
BIC: GENODEM1GLS

Vielen Dank für Ihre Aufmerksamkeit!