LS4-5: SQL-DQL (Aggregation-Gruppierung-Sortierung) - LÖSUNG

# Situation

Die Datenbank für den Online-Shop der Höllental-Senfmühle ist nun fertig gestellt und alle Daten migriert. Bevor wir den Auftrag abschließen und die Datenbank an den Auftraggeber übergeben, stehen noch etwaige Funktionstests an.

Liebe Kolleginnen und Kollegen, Sie haben bisher gut gearbeitet und wir konnten den Projektplan zur Konfiguration der Datenbank für den Online-Shop der Höllental-Senfmühle einhalten. Bevor wir nun den Auftrag abschließen, müssen noch diverse Funktionstests durchgeführt werden, um sicherzustellen, dass die Datenbank korrekt konfiguriert ist. Nehmen Sie bitte das Testprotokoll zur Hand und kontrollieren Sie die definierten Lösungen.

Welches Problem/Aufgabe stellt sich dar?

Welche Lösungsmöglichkeit(n) besteht(en)?

# Handlungsaufträge

1. Informieren Sie sich über die SQL-Befehle der ‚DQL‘ in Bezug auf Aggregation, Gruppierung und Sortierung.

Funktionstest zu Abfragen durchführen

2. Bearbeiten Sie die Datenbank der Höllental-Senfmühle und führen Sie die SQL-Befehle aus.

SQL-DQL

Daten abfragen – TESTPROTOKOLL I

Testen Sie bitte folgende Abfragen und dokumentieren Sie Ihr Testergebnis:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Testergebnis | 🞏 Fehlerfrei  🞏 Fehler:  \_\_\_\_\_\_\_\_\_\_\_\_ | 🞏 Fehlerfrei  🞏 Fehler:  \_\_\_\_\_\_\_\_\_\_\_\_ | 🞏 Fehlerfrei  🞏 Fehler:  \_\_\_\_\_\_\_\_\_\_\_\_ | 🞏 Fehlerfrei  🞏 Fehler:  \_\_\_\_\_\_\_\_\_\_\_\_ | 🞏 Fehlerfrei  🞏 Fehler:  \_\_\_\_\_\_\_\_\_\_\_\_ |
| Zu erwartendes Ergebnis | 41,97 € | 109,28 € | 7,55 € |  |  |
| Abfrage | Ermitteln Sie den durchschnittlichen Einkaufspreis aller Artikel.  SELECT AVG(Einkaufspreis) AS 'Anzahl'  FROM artikel; | Ermitteln Sie den maximalen Einkaufspreis aller Artikel.  SELECT MAX(Einkaufspreis) AS 'Anzahl'  FROM artikel; | Ermitteln Sie den minimalen Einkaufspreis aller Artikel.  SELECT MIN(Einkaufspreis) AS 'Anzahl'  FROM artikel; | Ermitteln Sie die Anzahl pro Warengruppe und geben Sie diese gruppiert nach der Warengruppe aus.  SELECT warengruppe\_id, COUNT(\*) AS 'Anzahl'  FROM artikel  GROUP BY warengruppe\_id; | Ermitteln Sie eine Liste, in der alle Artikel absteigend nach dem Einkaufspreis sortiert wird.  SELECT \*  FROM artikel  ORDER BY Einkaufspreis DESC; |

SQL – Aggregation-Gruppierung-Sortierung - INFO

# SQL – Aggregation (DQL)

Mit Hilfe von Aggregationsfunktionen können alle Datensätze (= mehrere Zeilen) einer Ergebnismenge zu einem einzigen Datensatz (= eine Zeile) zusammengefasst werden. Die Abfragen können sowohl ein als auch mehrere Merkmale (Spalten) betreffen. Wenn dabei Aggregationsfunktionen verwendet werden, muss für jedes Merkmal separat eine Aggregationsfunktion bestimmt werden.

Syntax Aggregatsfunktionen:

SELECT attribut [, AGGREGATSFUNKTION]

FROM tabellenname;

Folgende Aggregationsfunktionen unterstützt SQL:

|  |  |
| --- | --- |
| Anweisungsbefehl | Einsatzzweck |
| MIN (*Merkmalname*) | Minimalwert einer Spalte ausgeben |
| MAX (*Merkmalname*) | Maximalwert einer Spalte ausgeben |
| COUNT(\*) | Anzahl der Zeilen einer Tabelle ausgeben |
| COUNT(*Merkmalname*) | Anzahl der Zeilen, die bei dem Merkmal nicht den Wert NULL haben |
| COUNT(DISTINCT *Merkmalname*) | Anzahl der Zeilen mit unterschiedlichen Werten |
| SUM(*Merkmalname*) | Summenwert einer Spalte berechnen |
| AVG(*Merkmalname*) | Durchschnittswert einer Spalte berechnen |

Bei der Verwendung von **Aggregationsfunktionen im SELECT-Teil** einer SQL-Abfrage entstehen Spalten mit neuen bzw. veränderten Bedeutungen. Diesen Spalten können mit dem Schlüsselwort **AS** eine sinnvolle Spaltenüberschrift gegeben werden (**⭢ AS** ‚**Aliasnamen‘**).

Beispiel

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Tabelle Artikel* | |  |  |  |  |  |  |
| **# A\_Nr** | **Bezeichnung** | **Liefereinheit in Stk** | **Einzelpreis** | **Lagerbestand** | **Bestelleinheit** | **Lieferant** | **Kategorie** |
| 1 | Gummibärchen | 100 | 15,95 | 15 | 5 | 2 | 1 |
| 2 | Schokolade | 100 | 22,90 | 56 | 0 | 2 | 1 |
| 3 | Zucker | 24 | 17,99 | 22 | 15 | 2 | 5 |
| 4 | Pasta | 24 | 14,99 | 41 | 0 | 1 | 4 |
| 5 | Camembert | 15 | 12,35 | 12 | 25 | 4 | 2 |
| 6 | Bratwürste | 50 | 75,90 | 2 | 10 | 3 | 3 |

* **Funktion MIN: Wie hoch ist unser geringster Lagerbestand?**

|  |  |  |
| --- | --- | --- |
| ***Statement*** |  | ***Ergebnis*** |
| SELECT MIN(Lagerbestand) AS `Geringster Lagerbestand`  FROM Artikel; |  | **Geringster Lagerbestand** |
|  | 2 |

* **Funktion MAX: Welcher Artikel hat den alphabetisch letzten Namen?**

|  |  |  |
| --- | --- | --- |
| ***Statement*** |  | ***Ergebnis*** |
| SELECT MAX(Bezeichnung) AS `Letzter`  FROM Artikel; |  | **Letzter** |
|  | Zucker |

* **Funktion COUNT: Wie viele Artikel gibt es?**

|  |  |  |
| --- | --- | --- |
| ***Statement*** |  | ***Ergebnis*** |
| SELECT COUNT(\*) AS `Anzahl`  FROM Artikel; |  | **Anzahl** |
|  | 6 |

* **Funktion COUNT(DISTINCT): Wie viele unterschiedliche Kategorien gibt es?**

|  |  |  |
| --- | --- | --- |
| ***Statement*** |  | ***Ergebnis*** |
| SELECT COUNT(DISTINCT Kategorie) AS `Anzahl Kat`  FROM Artikel; |  | **Anzahl Kat** |
|  | 5 |

* **Funktion SUM: Wie hoch ist die gesamte, bestellte Einheit bei Lieferant Nr. 2?**

|  |  |  |
| --- | --- | --- |
| ***Statement*** |  | ***Ergebnis*** |
| SELECT SUM(Bestelleinheit) AS `Bestellmenge`  FROM Artikel  WHRE Lieferant = `2`; |  | **Bestellmenge** |
|  | 20 |
|  |  |

* **Funktion AVG: Wie hoch ist der durchschnittliche Einzelpreis aller Artikel?**

|  |  |  |
| --- | --- | --- |
| ***Statement*** |  | ***Ergebnis*** |
| SELECT AVG(Einzelpreis) AS `Durchschnittspreis`  FROM Artikel; |  | **Durchschnittspreis** |
|  | 26,68 |

# SQL – Gruppierung (DQL)

|  |  |  |
| --- | --- | --- |
| Mit dem SQL-Befehl **GROUP BY** können ausgewählte Daten gruppiert/zusammengefasst werden. |  | Wenn nicht alle Gruppen angezeigt werden sollen, kann die Auswahl durch den Befehl **HAVING[[1]](#footnote-2)** eingeschränkt werden |
| **Syntax GROUP BY:**  SELECT attribut [, AGGREGAT]  FROM tabellenname  [GROUP BY spaltenname]; |  | **Syntax GROUP BY HAVING:**  SELECT attribut [, AGGREGAT]  FROM tabellenname  [GROUP BY spaltenname]  [HAVING bedingung]; |

Beispiel

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Tabelle Artikel* | |  |  |  |  |
| **# A\_Nr** | **A\_Bez** | **A\_Art** | **A\_VK** | **A\_Bestand** | **A\_MinBestand** |
| 1 | HP Laserjet P6 | Laserdrucker | 675,00 | 15 | 2 |
| 2 | Cannon BPL85 | Laserdrucker | 580,00 | 4 | 1 |
| 3 | Macom 20 Zoll | Monitor | 310,00 | 10 | 4 |
| 4 | HP 24 Zoll | Monitor | 250,00 | 7 | 5 |
| 5 | Logitec HL20 | Tastatur | 119,00 | 20 | 5 |
| 6 | Cherry BT5 | Tastatur | 59,00 | 15 | 4 |
| 7 | Logitec Game5 | Tastatur | 189,00 | 8 | 4 |

* **Ermitteln Sie die Anzahl der Datensätze aller Artikelarten.**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Statement*** |  | ***Ergebnis*** |  |
| SELECT A\_Art, COUNT(\*) AS `Anzahl`  FROM Artikel  **GROUP BY A\_Art**; |  | **A\_Art** | **Anzahl** |
|  | Tastatur | 3 |
|  | Laserdrucker | 2 |
|  | Monitor | 2 |

* **Ermitteln Sie die Artikelgruppen, deren Durchschnittspreis über 200 € liegt.**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Statement*** |  | ***Ergebnis*** |  |
| SELECT A\_Art, COUNT(\*) AS `Anzahl`  FROM Artikel  **GROUP BY A\_Art**  **HAVING AVG(A\_VK)> 200**; |  | **A\_Art** | **Anzahl** |
|  | Laserdrucker | 2 |
|  | Monitor | 2 |
|  |  |  |

# SQL – ORDER BY (DQL)

Um sich die Ausgabe sortiert anzeigen zu lassen, wird der Befehl ORDER BY verwendet. Die Reihenfolge der Sortierung kann mit ASC (ascending) aufsteigend oder DESC (descending) absteigend beeinflusst werden. Wird weder DESC noch ASC als Sortierreihenfolge angegeben, kommt per Default ASC zum Zuge. ORDER BY steht als allerletzte Zeile in der SQL-Abfrage.

Syntax ORDER BY:

SELECT attribut [AGGREGATSFUNKTION]

FROM tabellenname;

[WHERE bedingung]

[GROUP BY spaltenname]

[HAVING bedingung]

[ORDER BY spaltenname [ASC/DESC]];

Beispiel

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Tabelle Kunde* | |  |  |  |  |  |
| **# K\_Nr** | **K\_Name** | **K\_Vorname** | **K\_Strasse** | **K\_PLZ** | **K\_Ort** | **K\_Telefon** |
| 1 | Schwab | Christoph | Fasanenweg 5 | 90600 | Nurnberg | NULL |
| 2 | Wagner | Sabine | Kanalstr. 26 | 90844 | Nurnberg | 0911-58763 |
| 3 | Meiler | Egon | Schulstr. 17 | 90780 | Fuerth | 0911-99598 |
| 4 | Schaller | Otto | Ulmenallee 46 | 90450 | Nurnberg | 0911-70403 |
| 5 | Huber | Xaver | Maktplatz 8 | 8900 | Munchen | 089-443322 |
| 6 | Lohmeier | Karin | Hohlweg 18 | 89445 | Munchen | 089-44667 |

* **Lassen Sie sich die gesamte Kundentabelle nach dem Nachnamen aufsteigend anzeigen.**

|  |
| --- |
| ***Statement*** |
| SELECT \*  FROM Kunde  ORDER BY K\_Name; | |
|

Ergebnis

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Tabelle Kunde* | |  |  |  |  |  |
| **# K\_Nr** | **K\_Name** | **K\_Vorname** | **K\_Strasse** | **K\_PLZ** | **K\_Ort** | **K\_Telefon** |
| 5 | Huber | Xaver | Maktplatz 8 | 8900 | Munchen | 089-443322 |
| 6 | Lohmeier | Karin | Hohlweg 18 | 89445 | Munchen | 089-44667 |
| 3 | Meiler | Egon | Schulstr. 17 | 90780 | Fuerth | 0911-99598 |
| 4 | Schaller | Otto | Ulmenallee 46 | 90450 | Nurnberg | 0911-70403 |
| 1 | Schwab | Christoph | Fasanenweg 5 | 90600 | Nurnberg | NULL |
| 2 | Wagner | Sabine | Kanalstr. 26 | 90844 | Nurnberg | 0911-58763 |

1. Hinweis: Die HAVING-Klausel wird nach dem GROUP BY-Befehl ausgeführt. Es ist deshalb zu empfehlen, alle Einschränkungen vorher zu regeln, z. B. als Teil von WHERE-Bedingungen. Nur wenn diese Einschränkung erst am Schluss geprüft werden kann, ist HAVING zu benutzen. HAVING entspricht damit einer WHERE-Bedingung bezogen auf einen GROUP-BY-Befehl. [↑](#footnote-ref-2)