Komplexe SQL-Anfragen

SQL-Anfragen mit komplexer Selektion, Subqueries, Gruppierung und Aggregationsfunktionen

Wichtigste Schlüsselworte (neben SELECT, FROM, und WHERE) in SQL im Bereich Subqueries (Unteranfragen), Gruppierung und Aggregationsfunktionen sind:

- **IN, ANY, ALL, EXISTS**
- **GROUP BY, HAVING**
- **COUNT, MIN, MAX, AVG, SUM**
- **DISTINCT**

Die Beispiele in diesem Material beziehen sich auf die Tabelle customers aus der W3C SQL School (Google: w3c sql school). Alle Anfragen lassen sich in der W3C SQL School unter SQL-Try-It ausführen. Durch die Eingabe von leicht veränderten Versionen der Anfragen kann man den Einfluss der verschiedenen Konstrukte nachvollziehen. Z.B. lässt sich

- **die SELECT-Klausel durch Hinzunehmen oder Entfernen von Spalten,**
- **die FROM-Klausel durch Hinzunehmen oder Entfernen von Tupelvariablen (Alias-Angaben) für die Tabelle,**
- **die WHERE- und HAVING-Klausel durch Modifikation der Formel (z.B. Austauschen von Konstanten, Austauschen von Vergleichoperatoren, Vertauschen von AND und OR, Vertauschen von ANY und ALL),**
- **die GROUP BY-Klausel durch Hinzunehmen oder Entfernen von Spalten und**
- **die ORDER BY-Klausel durch Vertauschen der Spaltenreihenfolge und Hinzunehmen oder Entfernen von ASC oder DESC**

verändern und das neue Ergebnis mit dem ursprünglichen Ergebnis vergleichen.

SELECT, FROM

```
SELECT *
FROM customers
```

CustomerID	CompanyName	ContactName	Address	City	PostalCode	Country
ALFKI	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
BERGS	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
CENTC	Centro comercial Moctezuma	Francisco Chang	Sierras de Granada 9993	México D.F.	05022	Mexico
ERNSH	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	8010	Austria
FISSA	FISSA Fabrica Inter. Salchichas S.A.	Diego Roel	C/ Moralzarzal, 86	Madrid	28034	Spain
GALED	Galería del gastrónomo	Eduardo Saavedra	Rambla de Cataluña, 23	Barcelona	08022	Spain
ISLAT	Island Trading	Helen Bennett	Garden House Crowther Way	Cowes	PO31 7PJ	UK
KOENE	Königlich Essen	Philip Cramer	Maubelstr. 90	Brandenburg	14776	Germany
LAUGB	Laughing Bacchus Wine Cellars	Yoshi Tannamuri	1900 Oak St.	Vancouver	V3F 2K1	Canada
MAGAA	Magazzini Alimentari Riuniti	Giovanni Rovelli	Via Ludovico il Moro 22	Bergamo	24100	Italy
NORTS	North/South	Simon Crowther	South House 300 Queensbridge	London	SW7 1RZ	UK
PARIS	Paris spécialités	Marie Bertrand	265, boulevard Charonne	Paris	75012	France
RATTC	Rattlesnake Canyon Grocery	Paula Wilson	2817 Milton Dr.	Albuquerque	87110	USA

SIMOB	Simons bistro	Jytte Petersen	Vinbæltet 34	København	1734	Denmark
THEBI	The Big Cheese	Liz Nixon	89 Jefferson Way Suite 2	Portland	97201	USA
VAFFE	Vaffeljernet	Palle Ibsen	Smagsløget 45	Århus	8200	Denmark
WOLZA	Wolski Zajazd	Zbyszek Piestrzeniewicz	ul. Filtrowa 68	Warszawa	01-012	Poland

WHERE, ORDER BY

```
SELECT Country
FROM customers
WHERE Country<='Italy'
ORDER BY Country
```

Country
Austria
Canada
Denmark
Denmark
France
Germany
Germany
Italy

SELECT DISTINCT

```
SELECT DISTINCT Country
FROM customers
WHERE Country<='Italy'
ORDER BY Country
```

Country
Austria
Canada
Denmark
France
Germany
Italy

AND, OR, NOT (Konjunktion, Disjunktion, Negation)

```
SELECT CompanyName, Country, PostalCode
FROM customers
WHERE Country IN ('Germany','Austria','Switzerland')

SELECT CompanyName, Country, PostalCode
FROM customers
WHERE Country='Germany' OR Country='Austria' OR Country='Switzerland'
```

CompanyName	Country	PostalCode
Alfreds Futterkiste	Germany	12209
Ernst Handel	Austria	8010
Königlich Essen	Germany	14776

```
SELECT CompanyName, Country, PostalCode
FROM customers
WHERE LEN(PostalCode)=4

SELECT CompanyName, Country, PostalCode
FROM customers
WHERE PostalCode LIKE '____'

'____' = 4 x Unterstrich
```

CompanyName	Country	PostalCode
Ernst Handel	Austria	8010
Simons bistro	Denmark	1734
Vaffeljernet	Denmark	8200

```
SELECT CompanyName, Country, PostalCode
FROM customers
WHERE Country IN ('Germany','Austria','Switzerland')
      AND
      LEN(PostalCode)=4
```

Bemerkung zum Layout von Anfragen: Alle Anfragen sollen aus diesem Material kopiert werden und in der W3C SQL School ausprobiert werden können. Daher muss der HTML-Text der Anfragen einfach sein und kann teilweise leider, zumindest was das Layout betrifft, nicht optimal gestaltet werden.

CompanyName	Country	PostalCode
Ernst Handel	Austria	8010

```
SELECT CompanyName, Country, PostalCode
FROM customers
WHERE Country IN ('Germany','Austria','Switzlerland)
      OR
      LEN(PostalCode)=4
```

OR entspricht der Mengenvereinigung von Anfrageergebnissen, AND entspricht dem Mengendurchschnitt.

CompanyName	Country	PostalCode
Alfreds Futterkiste	Germany	12209
Ernst Handel	Austria	8010
Königlich Essen	Germany	14776
Simons bistro	Denmark	1734
Vaffeljernet	Denmark	8200

Obacht bei der Ersetzung des IN-Terms durch den OR-Term in der AND-Anfrage: Diese Anfrage liefert 3 Tupel, während die ursprüngliche AND-Anfrage 1 Tupel lieferte. Die Klammersetzung muss beachtet werden. In der Anfrage bindet das AND stärker als das OR.

```
SELECT CompanyName, Country, PostalCode
FROM customers
WHERE Country='Germany'
      OR Country='Austria'
      OR Country='Switzerland'
      AND
      LEN(PostalCode)=4
```

CompanyName	Country	PostalCode
Alfreds Futterkiste	Germany	12209
Ernst Handel	Austria	8010
Königlich Essen	Germany	14776

```
SELECT CompanyName, Country, PostalCode
FROM customers
WHERE Country='Germany'
      OR Country='Austria'
      OR (Country='Switzerland'
          AND
          LEN(PostalCode)=4)

SELECT CompanyName, Country, PostalCode
FROM customers
WHERE (Country='Germany'
       OR Country='Austria'
       OR Country='Switzerland')
      AND
      LEN(PostalCode)=4
```


Verwendung von NOT:

```
SELECT CompanyName, Country, PostalCode
FROM customers
WHERE NOT ( Country IN ('Germany','Austria','Switzerland') )

SELECT CompanyName, Country, PostalCode
FROM customers
WHERE Country NOT IN ('Germany','Austria','Switzerland')
```

CompanyName	Country	PostalCode
Berglunds snabbköp	Sweden	S-958 22
Centro comercial Moctezuma	Mexico	05022
FISSA Fabrica Inter. Salchichas S.A.	Spain	28034
Galería del gastrónomo	Spain	08022
Island Trading	UK	PO31 7PJ
Laughing Bacchus Wine Cellars	Canada	V3F 2K1
Magazzini Alimentari Riuniti	Italy	24100
North/South	UK	SW7 1RZ
Paris spécialités	France	75012
Rattlesnake Canyon Grocery	USA	87110
Simons bistro	Denmark	1734
The Big Cheese	USA	97201
Vaffeljernet	Denmark	8200
Wolski Zajazd	Poland	01-012

Beobachtung: 'Switzerland' taucht weder im Datenbankzustand unter Country noch im Ergebnis der Anfrage auf.

NOT entspricht der Komplementbildung von Anfrageergebnissen bzgl. der gespeicherten Tabelle.

Gesetze der Logik

Kommutativität

- **F AND G = G AND F**
- **F OR G = G OR F**

Assoziativität

- **(F AND G) AND H = F AND (G AND H) = F AND G AND H**
- **(F OR G) OR H = F OR (G OR H) = F OR G OR H**

Distributivität

- **F AND (G OR H) = (F AND G) OR (F AND H) = F AND G OR F AND H**
- **F OR (G AND H) = (F OR G) AND (F OR H)**

DeMorgan

- **NOT (F AND G) = (NOT F) OR (NOT G)**
- **NOT (F OR G) = (NOT F) AND (NOT G)**

Weitere Gesetze siehe http://de.wikipedia.org/wiki/Boolesche_Algebra.

Diese Gesetze sind in SQL anwendbar. Idealerweise berücksichtigt ein Anfrageauswerter und -optimierer diese Gesetze. Dies ist jedoch nicht immer der Fall. Somit können diese Gesetze auch zur Umformulierung dienen und damit möglicherweise zu einer schnelleren Anfragebearbeitung führen.

Disjunktive und Konjunktive Normalform (Logik)

Bei der disjunktiven Normalform (DNF) handelt es sich um einen logischen Ausdruck, der aus ODER-Verknüpfungen besteht. Der logische Ausdruck besteht in der obersten Ebene ausschließlich aus ODER-Verknüpfungen. Dabei können die einzelnen Elemente der ODER-Verknüpfung kompliziertere Ausdrücke sein, die dann auch eine UND-Verknüpfung

enthalten können. (http://de.wikipedia.org/wiki/Disjunktive_Normalform)

Bei der konjunktiven Normalform (KNF) handelt es sich um einen logischen Ausdruck, der aus UND-Verknüpfungen besteht. Der logische Ausdruck besteht in der obersten Ebene ausschließlich aus UND-Verknüpfungen. Dabei können die einzelnen Elemente der UND-Verknüpfung kompliziertere Ausdrücke sein, die dann auch eine ODER-Verknüpfung enthalten können.

1. Anfrage in DNF

```
SELECT Country, City
FROM customers
WHERE ('I'<=Country AND Country<='P') OR
      ('I'<=City AND City<='P')
ORDER BY Country, City
```

Äquivalente Anfrage in KNF

```
SELECT Country, City
FROM customers
WHERE ('I'<=Country OR 'I'<=City) AND
      ('I'<=Country OR City<='P') AND
      (Country<='P' OR 'I'<=City) AND
      (Country<='P' OR City<='P')
ORDER BY Country, City
```

	Z | P	++++++++++++ ++++++++++++ ++++++++++++	----------	++++++++++++ ++++++++++++ ++++++++++++
City	P | I	++++++++++++ ++++++++++++ ++++++++++++	++++++++++++ ++++++++++++ ++++++++++++	++++++++++++ ++++++++++++ ++++++++++++
	I | A	++++++++++++ ++++++++++++ ++++++++++++	----------	++++++++++++ ++++++++++++ ++++++++++++
		A _ I	I _ P	P _ Z
			Country	

2. Anfrage in KNF (nicht äquivalent zur 1. Anfrage)

```
SELECT Country, City
FROM customers
WHERE ('I'<=Country OR Country<='P') AND
      ('I'<=City OR City<='P')
ORDER BY Country, City
```

Äquivalente Anfrage in DNF

```
SELECT Country, City
FROM customers
WHERE ('I'<=Country AND 'I'<=City) OR
      ('I'<=Country AND City<='P') OR
      (Country<='P' AND 'I'<=City) OR
      (Country<='P' AND City<='P')
ORDER BY Country, City
```

Obacht: Diese Anfrage ist durch Vertauschen von AND und OR aus der 1. Anfrage hervorgegangen. Man beachte den Einfluss der WHERE-Klausel auf die Resultatsmenge.

3. Anfrage in KNF (nicht äquivalent zur 1. und 2. Anfrage)

```
SELECT Country, City
FROM customers
WHERE (Country<='I' OR 'P'<=Country) AND
      (City<='I' OR 'P'<=City)
ORDER BY Country, City
```

Äquivalente Anfrage in DNF

```
SELECT Country, City
FROM customers
WHERE (Country<='I' AND City<='I') OR
      (Country<='I' AND 'P'<=City) OR
      ('P'<=Country AND City<='I') OR
      ('P'<=Country AND 'P'<=City)
ORDER BY Country, City
```

	Z | P	++++++++++++ ++++++++++++ ++++++++++++	----------	++++++++++++ ++++++++++++ ++++++++++++
City	P | I	----------	----------	----------
	I | A	++++++++++++ ++++++++++++ ++++++++++++	----------	++++++++++++ ++++++++++++ ++++++++++++
		A _ I	I _ P	P _ Z
			Country	

BETWEEN

```
SELECT CompanyName, Country
FROM customers
WHERE Country BETWEEN 'France' AND 'Poland'
ORDER BY Country

SELECT CompanyName, Country
FROM customers
WHERE 'France'<=Country AND Country<='Poland'
ORDER BY Country
```

CompanyName	Country
Paris spécialités	France
Königlich Essen	Germany
Alfreds Futterkiste	Germany
Magazzini Alimentari Riuniti	Italy
Centro comercial Moctezuma	Mexico
Wolski Zajazd	Poland

LEFT, INSTR, RIGHT, LEN, AS

Obacht: LEFT, INSTR, RIGHT und LEN sind im Standard anders benannt.

Standard-SQL: CHAR_LENGTH(arg) liefert die Länge.

SUBSTRING(arg FROM pos) liefert Teilzeichenkette ab Position pos.

SUBSTRING(arg FROM pos FOR length) liefert Teilzeichenkette ab Position pos in der Länge length.

POSITION(part IN arg) liefert die Position die die Teilzeichenkette part in der Zeichenkette arg besitzt.

```
SELECT
  LEFT(ContactName,INSTR(ContactName,' ')-1)
    AS FirstName,
  RIGHT(ContactName,LEN(ContactName)-INSTR(ContactName,' '))
    AS LastName
FROM customers
WHERE Country<='Italy'
ORDER BY 2, 1
```

FirstName	LastName
Maria	Anders
Marie	Bertrand
Philip	Cramer
Palle	Ibsen

Roland	Mendel
Jytte	Petersen
Giovanni	Rovelli
Yoshi	Tannamuri

LIKE

```
SELECT ContactName, CompanyName
FROM customers
WHERE LEFT(ContactName,INSTR(ContactName,' ')-1) LIKE '%a'
ORDER BY 1
```

ContactName	CompanyName
Christina Berglund	Berglunds snabbköp
Maria Anders	Alfreds Futterkiste
Paula Wilson	Rattlesnake Canyon Grocery

```
SELECT CompanyName, City
FROM customers
WHERE City LIKE '%burg'
ORDER BY 1
```

CompanyName	City
Königlich Essen	Brandenburg

```
SELECT CompanyName, PostalCode
FROM customers
WHERE Country='Germany' AND PostalCode LIKE '1____'
ORDER BY 1

'____' = 4 x Unterstrich
```

CompanyName	PostalCode
Alfreds Futterkiste	12209
Königlich Essen	14776

Im allgemeinen nicht äquivalent zur obigen Anfrage, hier jedoch das gleiche Ergebnis liefernd, ist:

```
SELECT CompanyName, PostalCode
FROM customers
WHERE Country='Germany' AND PostalCode LIKE '1%'
ORDER BY 1
```

'1____' testet auf eine '1' am Anfang und 4 folgende Zeichen; '1%' testet auf eine '1' am Anfang und beliebig viele folgende Zeichen; z.B., ergibt '1%' angewendet auf '1' den Wert TRUE und '1____' angewendet auf '1' liefert FALSE.

Exotische Vornamen (Vornamen, die die Buchstaben X, Y oder Z enthalten)

```
SELECT LEFT(ContactName,INSTR(ContactName,' ')-1) AS FirstName
FROM customers
WHERE LEFT(ContactName,INSTR(ContactName,' ')-1) LIKE '%X%' OR
      LEFT(ContactName,INSTR(ContactName,' ')-1) LIKE '%x%' OR
      LEFT(ContactName,INSTR(ContactName,' ')-1) LIKE '%Y%' OR
      LEFT(ContactName,INSTR(ContactName,' ')-1) LIKE '%y%' OR
      LEFT(ContactName,INSTR(ContactName,' ')-1) LIKE '%Z%' OR
      LEFT(ContactName,INSTR(ContactName,' ')-1) LIKE '%z%'
ORDER BY  1
```

FirstName
Jytte
Liz
Yoshi
Zbyszek

<=All-Subquery

Lexikographisch kleinster PostalCode

```
SELECT Country, PostalCode
FROM customers
WHERE PostalCode <=ALL (SELECT PostalCode
                        FROM customers)
```

Country	PostalCode
Poland	01-012

MIN-Abfrage mit Ein-Spalten-Selektion

```
SELECT Country, PostalCode
FROM customers
WHERE PostalCode=(SELECT MIN(PostalCode)
                  FROM customers)
```

Die Konstruktion einer Unteranfrage mit ALL bietet im allgemeinen Fall mehr Möglichkeiten. Nicht alle mit ALL formulierten Unteranfragen lassen sich auf Aggregationsfunktionen zurückführen.

<>All-Subquery

```
SELECT Country, PostalCode
FROM customers
WHERE PostalCode <>ALL (SELECT c2.PostalCode
                        FROM customers c1, customers c2
                        WHERE c1.PostalCode<c2.PostalCode)
```

Die Unteranfrage berechnet die Menge der Postleitzahlen c2.PostalCode, zu denen es eine Postleitzahl c1.PostalCode gibt, die lexikographisch vor der Ergebnis-Postleitzahl c2.PostalCode liegt. Die Unteranfrage berechnet die Menge der 'lexikographisch grösseren' Postleitzahlen.

Kartesisches Produkt (Kreuzprodukt) von Tabellen

```
SELECT *
FROM customers c1, customers c2
```

Werden mehr als eine Tabelle in der FROM-Klausel genannt und die Tabellen inklusive möglicher Alias-Angaben (Tupel- oder Satz-Variablen) durch Kommata getrennt, wird das Kreuzprodukt der angegebenen Tabellen berechnet. D.h., jeder Satz aus der ersten Tabelle wird mit jedem Satz aus der zweiten Tabelle kombiniert. Im Beispiel ergibt sich dann eine Resultatsmenge mit 7+7=14 Spalten und 17*17=289 Sätzen (Zeilen,Tupeln).

```
SELECT *
FROM customers c1, customers c2
WHERE c1.CompanyName<c2.CompanyName AND
      c1.ContactName<c2.ContactName AND
      c1.Address<c2.Address AND
      c1.City<c2.City AND
      c1.PostalCode<c2.PostalCode AND
      c1.Country<c2.Country
```

c1.CustomerID	c1.CompanyName	...	...	...	...	...	c2.CustomerID	c2.CompanyName	...	...	...	...	...
ALFKI	Alfreds Futterkiste						NORTS	North/South					
ERNSH	Ernst Handel						NORTS	North/South					
GALED	Galería del gastrónomo						NORTS	North/South					
KOENE	Königlich Essen						NORTS	North/South					

```
SELECT c1.CompanyName, c2.CompanyName
FROM customers c1, customers c2
WHERE c1.CompanyName<c2.CompanyName AND
```

```
      c1.ContactName<c2.ContactName AND
      c1.Address<c2.Address AND
      c1.City<c2.City AND
      c1.PostalCode<c2.PostalCode AND
      c1.Country<c2.Country
```

c1.CompanyName	c2.CompanyName
Alfreds Futterkiste	North/South
Ernst Handel	North/South
Galería del gastrónomo	North/South
Königlich Essen	North/South

=ANY-Subquery

```
SELECT Country, PostalCode
FROM customers
WHERE NOT ( PostalCode =ANY (SELECT c2.PostalCode
                             FROM customers c1, customers c2
                             WHERE c1.PostalCode<c2.PostalCode) )
```

IN-Subquery

```
SELECT Country, PostalCode
FROM customers
WHERE NOT ( PostalCode IN (SELECT c2.PostalCode
                           FROM customers c1, customers c2
                           WHERE c1.PostalCode<c2.PostalCode) )
```

EXISTS-Subquery

```
SELECT Country, PostalCode
FROM customers c
WHERE NOT ( EXISTS (SELECT c2.PostalCode
                    FROM customers c1, customers c2
                    WHERE c.PostalCode=c2.PostalCode AND
                          c1.PostalCode<c2.PostalCode) )

SELECT Country, PostalCode
FROM customers c
WHERE NOT ( EXISTS (SELECT *
                    FROM customers c1, customers c2
                    WHERE c.PostalCode=c2.PostalCode AND
                          c1.PostalCode<c2.PostalCode) )

SELECT Country, PostalCode
FROM customers c
WHERE NOT ( EXISTS (SELECT 42
                    FROM customers c1, customers c2
                    WHERE c.PostalCode=c2.PostalCode AND
                          c1.PostalCode<c2.PostalCode) )
```

Statt der Konstanten 42 kann eine beliebige andere Konstante angegeben werden, z.B. 4711, 3.14, 'ABBA', 'Zappa', FALSE oder TRUE.

Es sind nun mittlerweile 8 (in Worten acht) verschiedene äquivalente Formulierungen einer Anfrage angegeben worden. Weitere unterschiedliche Anfragetexte ergeben sich z.B. durch Anwendung des Kommutativgesetzes für AND.

Diese Unteranfragen im EXISTS können im Gegensatz zu den bisherigen Unteranfragen nicht direkt als Anfragen gestellt werden.

<=ALL-Subquery mit Korrelation

Lexikographisch kleinster PostalCode pro Land.

```
SELECT Country, PostalCode
FROM customers c
WHERE PostalCode <=ALL (SELECT PostalCode
```



```
                        FROM customers
                        WHERE c.Country=Country) AND
      Country<='Italy'
ORDER BY Country, PostalCode
```

Verwendung von Variablen nur an den Vorkommen wo die Variablen benötigt werden.

Country	PostalCode
Austria	8010
Canada	V3F 2K1
Denmark	1734
France	75012
Germany	12209
Italy	24100

Die erste obige Unteranfrage (SELECT PostalCode FROM customers) könnte auch direkt als Anfrage gestellt werden. Diese Unteranfrage hier könnte nicht als Anfrage gestellt werden, da die Variable c und damit der Ausdruck c.Country nicht bekannt sind.

Bezeichnung: Dies ist eine korrelierte Unteranfrage (Unteranfrage mit Korrelation versus Unteranfrage ohne Korrelation).

```
SELECT c_o.Country, c_o.PostalCode
FROM customers c_o
WHERE c_o.PostalCode <=ALL (SELECT c_i.PostalCode
                            FROM customers c_i
                            WHERE c_o.Country=c_i.Country) AND
      Country<='Italy'
ORDER BY Country, PostalCode
```

Verwendung von Variablen an allen Vorkommen der Variablen (c_o outer, c_i inner).

Analogie:

- Verwendung von Variablen in Anfragen: Variablen in blockorientierten Programmiersprachen
- { integer v_o; ... { integer v_i; ... } ... }
- SELECT ...
  FROM table v_o
  WHERE ... (SELECT ... FROM table v_i WHERE ...) ...
- Gültigkeitsbereich von v_o ist jeweils auch der innere Teil des Programms bzw. der Anfrage.

Falls keine Tupelvariablen deklariert werden, stehen die Namen der Tabellen als implizite Variablen zur Verfügung.

<ALL-Subquery

Obacht: Die Ersetzung von '<=All' durch '<ALL' in der obigen Anfrage ergibt *immer* die leere Resultatsmenge, unabhängig vom aktuellen Datenbankzustand.

```
SELECT Country, PostalCode
FROM customers c
WHERE PostalCode <ALL (SELECT PostalCode
                       FROM customers c_inner
                       WHERE c.Country=c_inner.Country)
ORDER BY Country, PostalCode
```

Country	PostalCode

NOT EXISTS-Subquery

```
SELECT c_o.Country, c_o.PostalCode
FROM customers c_o
WHERE NOT EXISTS (SELECT *
                  FROM customers c_i
                  WHERE c_o.Country=c_i.Country AND
                        c_o.PostalCode>c_i.PostalCode)
ORDER BY Country, PostalCode
```

MIN-Abfrage mit Ein-Spalten-Selektion

```
SELECT Country, PostalCode
FROM customers c
WHERE PostalCode=(SELECT MIN(PostalCode)
                  FROM customers
                  WHERE Country=c.Country)
ORDER BY Country, PostalCode
```

IN-Subquery

Länder, die mindestens zwei Sätze in der Tabelle customer besitzen, zusammen mit ihren Postleitzahlen.

```
SELECT Country, PostalCode
FROM customers
WHERE Country IN (SELECT Country
                  FROM customers
                  GROUP BY Country
                  HAVING COUNT(*)>=2)
ORDER BY Country, PostalCode
```

Country	PostalCode
Denmark	1734
Denmark	8200
Germany	12209
Germany	14776
Spain	08022
Spain	28034
UK	PO31 7PJ
UK	SW7 1RZ
USA	87110
USA	97201

=ANY-Subquery

```
SELECT Country, PostalCode
FROM customers
WHERE Country =ANY (SELECT Country
                    FROM customers
                    GROUP BY Country
                    HAVING COUNT(*)>=2)
ORDER BY Country, PostalCode
```

EXISTS-Subquery

```
SELECT Country, PostalCode
FROM customers c
WHERE EXISTS (SELECT *
              FROM customers c1, customers c2
              WHERE c.Country=c1.Country AND c.Country=c2.Country AND
                    c1.PostalCode<>c2.PostalCode)
ORDER BY Country, PostalCode
```

Beobachtung: Die Anfrage könnte auch mit GROUP BY/HAVING formuliert werden. Die Verwendung von GROUP BY/HAVING impliziert also nicht die zwingende Notwendigkeit der Verwendung dieser Sprachmittel.

=ALL-Subquery

```
SELECT Country, PostalCode
FROM customers c
WHERE LEN(PostalCode) =ALL (SELECT LEN(PostalCode)
                            FROM customers c_inner
                            WHERE c.Country=c_inner.Country)
ORDER BY Country, PostalCode
```


Country	PostalCode
Austria	8010
Canada	V3F 2K1
Denmark	1734
Denmark	8200
France	75012
Germany	12209
Germany	14776
Italy	24100
Mexico	05022
Poland	01-012
Spain	08022
Spain	28034
Sweden	S-958 22
USA	87110
USA	97201

Beobachtung: 'UK' nicht enthalten!

<>ANY-Subquery

```
SELECT Country, PostalCode
FROM customers c
WHERE LEN(PostalCode) <>ANY (SELECT LEN(PostalCode)
                             FROM customers c_inner
                             WHERE c.Country=c_inner.Country)
ORDER BY Country, PostalCode
```

Country	PostalCode
UK	PO31 7PJ
UK	SW7 1RZ

FORALL als NOT EXISTS NOT

SQL-92 kennt zwar eine EXISTS-Subquery und damit einen Existenzquantor für Tupel. Aber es gibt in SQL-92 (und damit in den meisten existierenden Systemen) keinen entsprechenden Allquantor. Der Allquantor kann jedoch mit Hilfe einer NOT EXISTS-Subquery ausgedrückt werden.

Erinnerung an Gesetze der Logik 1. Stufe

- **[für alle x gilt p(x)]**
  =
  es gilt nicht [es existiert x mit (es gilt nicht p(x))]
- **[es existiert x mit p(x)]**
  =
  es gilt nicht [für alle x gilt (es gilt nicht p(x))]

In SQL

- **FORALL (SELECT * FROM T WHERE F)**
  =
  NOT EXISTS (SELECT * FROM T WHERE NOT F)

Beispiel zu einer möglichen, teilweise hypothetischen Umformung einer SQL-Anfrage

```
SELECT Country, PostalCode
FROM customers c1
WHERE NOT EXISTS (SELECT *
                  FROM customers c2
                  WHERE c1.Country=c2.Country AND
                        c1.PostalCode>c2.PostalCode)

SELECT Country, PostalCode
FROM customers c1
WHERE NOT EXISTS (SELECT *
```

```
                FROM customers c2
                WHERE NOT(NOT(c1.Country=c2.Country AND
                              c1.PostalCode>c2.PostalCode)))

SELECT Country, PostalCode
FROM customers c1
WHERE NOT EXISTS (SELECT *
                  FROM customers c2
                  WHERE NOT(c1.Country<>c2.Country OR
                            c1.PostalCode<=c2.PostalCode))

SELECT Country, PostalCode -- KEIN GÜLTIGES SQL!
FROM customers c1
WHERE FORALL (SELECT *
              FROM customers c2
              WHERE c1.Country<>c2.Country OR
                    c1.PostalCode<=c2.PostalCode)

SELECT Country, PostalCode -- KEIN GÜLTIGES SQL!
FROM customers c1
WHERE FORALL (SELECT *
              FROM customers c2
              WHERE c1.Country=c2.Country IMPLIES
                    c1.PostalCode<=c2.PostalCode)
```

FEHLEN von IMPLIES und XOR in SQL

- IMPLIES und XOR fehlen in SQL, können aber ausgedrückt werden.
- F IMPLIES G = NOT(F) OR G
- F XOR G = (NOT(F) AND G) OR (F AND NOT(G))
- Die in SQL zu verwendenden Formulierungen sind aber häufig schlechter verständlich als eine Formulierung direkt mit IMPLIES oder XOR, insbesondere für XOR, da hier Teilterme dupliziert werden müssen.

GROUP BY, COUNT(*)

```
SELECT Country, COUNT(*) AS CountPostalCode
FROM customers
GROUP BY Country
ORDER BY 2,1
```

Country	CountPostalCode
Austria	1
Canada	1
France	1
Italy	1
Mexico	1
Poland	1
Sweden	1
Denmark	2
Germany	2
Spain	2
UK	2
USA	2

HAVING

```
SELECT Country, COUNT(*) AS CountPostalCode
FROM customers
GROUP BY Country
HAVING COUNT(*)>=2
ORDER BY 2,1
```

Country	CountPostalCode
Denmark	2
Germany	2

Spain	2
UK	2
USA	2

WHERE, HAVING

```
SELECT Country, COUNT(*) AS CountPostalCode
FROM customers
WHERE LEN(PostalCode)=5
GROUP BY Country
HAVING COUNT(*)>=2
ORDER BY 2,1
```

Country	CountPostalCode
Germany	2
Spain	2
USA	2

```
SELECT Country, PostalCode
FROM customers
WHERE LEN(PostalCode)>5
ORDER BY Country, PostalCode
```

Country	PostalCode
Canada	V3F 2K1
Poland	01-012
Sweden	S-958 22
UK	PO31 7PJ
UK	SW7 1RZ

```
SELECT Country,
       Count(*) AS CountPostalCode,
       AVG(LEN(PostalCode)) AS AvgPostalCodeLength
FROM customers
WHERE LEN(PostalCode)>5
GROUP BY Country
HAVING AVG(LEN(PostalCode))>7
```

Country	CountPostalCode	AvgPostalCodeLength
Sweden	1	8
UK	2	7.5

Integritätsbedingungen (IBen)

- Statische Integritätsbedingung (IB) bezieht sich auf *einen* Zustand (im Gegensatz zu dynamischen IBen, die sich auf mehr als einen Zustand beziehen, z.B. auf Zustandsübergänge oder Zustandsfolgen).
- Idealerweise formuliert als:
  FORALL (table_1 t_1) ... (table_n t_n) [p(t_1, ..., t_n)]
  Prädikat p über den Zeilen t_1, ..., t_n
- Realisierung durch Auffinden von Datensätzen, die der Integrität widersprechen.
- SELECT *
  FROM table_1 t_1 , ... , table_n t_n
  WHERE NOT p(t_1, ..., t_n)
- Das Ergebnis der entsprechenden Anfrage muss in einem gültigen Zustand die leere Menge ergeben.

Beispiele

- IB: Alle CustomerIDs sind fünfstellig.
- Finde alle Zeilen mit nicht-fünfstelligen CustomerID.

```
SELECT *
FROM customers
WHERE LEN(CustomerID)<>5
```

CustomerID	CompanyName	ContactName	Address	City	PostalCode	Country

- IB: Ein ContactName hat maximal ein Leerzeichen (ContactName hat Format: FirstName Leerzeichen LastName).
- Finde alle Zeilen in denen ein ContactName mindestens 2 Leerzeichen hat.

```
SELECT *
FROM customers
WHERE ContactName LIKE '% % %'
```

CustomerID	CompanyName	ContactName	Address	City	PostalCode	Country

- IB: PostalCode enthält mindestens eine Ziffer.
- Finde Zeilen mit PostalCodes die keine Ziffer enthalten.

```
SELECT *
FROM customers
WHERE PostalCode NOT LIKE '%0%' AND PostalCode NOT LIKE '%1%' AND
      PostalCode NOT LIKE '%2%' AND PostalCode NOT LIKE '%3%' AND
      PostalCode NOT LIKE '%4%' AND PostalCode NOT LIKE '%5%' AND
      PostalCode NOT LIKE '%6%' AND PostalCode NOT LIKE '%7%' AND
      PostalCode NOT LIKE '%8%' AND PostalCode NOT LIKE '%9%'

SELECT *
FROM customers
WHERE NOT (PostalCode LIKE '%0%' OR PostalCode LIKE '%1%' OR
           PostalCode LIKE '%2%' OR PostalCode LIKE '%3%' OR
           PostalCode LIKE '%4%' OR PostalCode LIKE '%5%' OR
           PostalCode LIKE '%6%' OR PostalCode LIKE '%7%' OR
           PostalCode LIKE '%8%' OR PostalCode LIKE '%9%')
```

CustomerID	CompanyName	ContactName	Address	City	PostalCode	Country

- IB: Ein deutscher PostalCode besteht aus genau 5 Ziffern.
- Finde alle Zeilen aus Deutschland mit einem PostalCode, der nicht aus genau 5 Ziffern besteht.

Folgende Anfrage verdeutlicht zunächst nur den Zugriff auf PostalCode.

```
SELECT PostalCode,
       LEFT(RIGHT(LEFT(PostalCode,5),5),1) AS PC1,
       LEFT(RIGHT(LEFT(PostalCode,5),4),1) AS PC2,
       LEFT(RIGHT(LEFT(PostalCode,5),3),1) AS PC3,
       LEFT(RIGHT(LEFT(PostalCode,5),2),1) AS PC4,
       LEFT(RIGHT(LEFT(PostalCode,5),1),1) AS PC5
FROM customers
WHERE Country='Germany' AND
      LEFT(RIGHT(LEFT(PostalCode,5),5),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,5),4),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,5),3),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,5),2),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,5),1),1) IN
        ('0','1','2','3','4','5','6','7','8','9')
```

PostalCode	PC1	PC2	PC3	PC4	PC5
12209	1	2	2	0	9
14776	1	4	7	7	6

Formulierung der IB:

```
SELECT *
FROM customers
```



```
WHERE Country='Germany' AND (
      LEFT(RIGHT(LEFT(PostalCode,5),5),1) NOT IN
        ('0','1','2','3','4','5','6','7','8','9') OR
      LEFT(RIGHT(LEFT(PostalCode,5),4),1) NOT IN
        ('0','1','2','3','4','5','6','7','8','9') OR
      LEFT(RIGHT(LEFT(PostalCode,5),3),1) NOT IN
        ('0','1','2','3','4','5','6','7','8','9') OR
      LEFT(RIGHT(LEFT(PostalCode,5),2),1) NOT IN
        ('0','1','2','3','4','5','6','7','8','9') OR
      LEFT(RIGHT(LEFT(PostalCode,5),1),1) NOT IN
        ('0','1','2','3','4','5','6','7','8','9') )
```

CustomerID	CompanyName	ContactName	Address	City	PostalCode	Country

- IB: CustomerID ist Primärschlüssel. Zwei unterschiedliche Zeilen besitzen unterschiedliche CustomerID-Werte.
- Finde alle Zeilen zu denen es eine weitere Zeile gibt, die den gleichen CustomerID hat, aber unterschiedliche Werte für CompanyName, ContactName, Address, City, PostalCode oder Country.

```
SELECT *
FROM customers c1
WHERE EXISTS (SELECT *
              FROM customers c2
              WHERE c1.CustomerId=c2.CustomerID AND
                    (c1.CompanyName<>c2.CompanyName OR
                     c1.ContactName<>c2.ContactName OR
                     c1.Address<>c2.Address OR
                     c1.City<>c2.City OR
                     c1.PostalCode<>c2.PostalCode OR
                     c1.Country<>c2.Country))
```

CustomerID	CompanyName	ContactName	Address	City	PostalCode	Country

Erzeugung eines 'virtuellen' Attributs 'Sales'

In customers fehlt ein numerisches Attribut an dem der Einsatz von Aggregationsfunktionen gezeigt werden kann. Daher wird ein solches Attribut (Sales, Umsatz) künstlich aus den (ersten vier Ziffern der) Postleitzahlen erzeugt.

```
SELECT Country, CompanyName, LEFT(PostalCode,4) AS Sales
FROM customers
WHERE LEFT(RIGHT(LEFT(PostalCode,4),4),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),3),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),2),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),1),1) IN
        ('0','1','2','3','4','5','6','7','8','9')
ORDER BY Country, CompanyName
```

Country	CompanyName	Sales
Austria	Ernst Handel	8010
Denmark	Simons bistro	1734
Denmark	Vaffeljernet	8200
France	Paris spécialités	7501
Germany	Alfreds Futterkiste	1220
Germany	Königlich Essen	1477
Italy	Magazzini Alimentari Riuniti	2410
Mexico	Centro comercial Moctezuma	0502
Spain	FISSA Fabrica Inter. Salchichas S.A.	2803
Spain	Galería del gastrónomo	0802
USA	Rattlesnake Canyon Grocery	8711
USA	The Big Cheese	9720

MIN, AVG, MAX

```
SELECT Country, MIN(LEFT(PostalCode,4)) AS MinSales,
       AVG(LEFT(PostalCode,4)) AS AvgSales,
       MAX(LEFT(PostalCode,4)) AS MaxSales
FROM customers
WHERE LEFT(RIGHT(LEFT(PostalCode,4),4),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),3),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),2),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),1),1) IN
        ('0','1','2','3','4','5','6','7','8','9')
GROUP BY Country
ORDER BY Country
```

Country	MinSales	AvgSales	MaxSales
Austria	8010	8010	8010
Denmark	1734	4967	8200
France	7501	7501	7501
Germany	1220	1348.5	1477
Italy	2410	2410	2410
Mexico	0502	502	0502
Spain	0802	1802.5	2803
USA	8711	9215.5	9720

```
SELECT Country, MIN(LEFT(PostalCode,4)) AS MinSales,
       AVG(LEFT(PostalCode,4)) AS AvgSales,
       MAX(LEFT(PostalCode,4)) AS MaxSales
FROM customers
WHERE LEFT(RIGHT(LEFT(PostalCode,4),4),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),3),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),2),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),1),1) IN
        ('0','1','2','3','4','5','6','7','8','9')
GROUP BY Country
ORDER BY 3 DESC
```

Country	MinSales	AvgSales	MaxSales
USA	8711	9215.5	9720
Austria	8010	8010	8010
France	7501	7501	7501
Denmark	1734	4967	8200
Italy	2410	2410	2410
Spain	0802	1802.5	2803
Germany	1220	1348.5	1477
Mexico	0502	502	0502

HAVING inklusive MIN- und AVG-Selektion

Länder mit kleiner Umsatzstreuung, d.h., Länder, bei denen der minimale Umsatz nur 20 Prozent unter dem durchschnittlichen Umsatz liegt. Ferner soll die Umsatzstreuung in dem betreffenden Land 'echt' sein in dem Sinne, dass es mehr als einen Umsatz gibt.

```
SELECT Country,
       MIN(LEFT(PostalCode,4)) AS MinSales,
       MIN(LEFT(PostalCode,4))*1.20 AS MinSalesTimes1Dot20,
       AVG(LEFT(PostalCode,4)) AS AvgSales,
       MAX(LEFT(PostalCode,4)) AS MaxSales
FROM customers
WHERE LEFT(RIGHT(LEFT(PostalCode,4),4),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),3),1) IN
```



```
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),2),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),1),1) IN
        ('0','1','2','3','4','5','6','7','8','9')
GROUP BY Country
HAVING COUNT(*)>1 AND
       MIN(LEFT(PostalCode,4))*1.20>=AVG(LEFT(PostalCode,4))
ORDER BY 1
```

Country	MinSales	MinSalesTimes1Dot20	AvgSales	MaxSales
Germany	1220	1464	1348.5	1477
USA	8711	10453.2	9215.5	9720

Wie oben ohne die einschränkende HAVING-Bedingung an MIN/AVG

```
SELECT Country,
       MIN(LEFT(PostalCode,4)) AS MinSales,
       MIN(LEFT(PostalCode,4))*1.20 AS MinSalesTimes1Dot20,
       AVG(LEFT(PostalCode,4)) AS AvgSales,
       MAX(LEFT(PostalCode,4)) AS MaxSales
FROM customers
WHERE LEFT(RIGHT(LEFT(PostalCode,4),4),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),3),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),2),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),1),1) IN
        ('0','1','2','3','4','5','6','7','8','9')
GROUP BY Country
HAVING COUNT(*)>1
ORDER BY 1
```

Country	MinSales	MinSalesTimes1Dot20	AvgSales	MaxSales
Denmark	1734	2080.8	4967	8200
Germany	1220	1464	1348.5	1477
Spain	0802	962.4	1802.5	2803
USA	8711	10453.2	9215.5	9720

MIN und MAX ohne Aggregationsfunktionen - Konstanten im SELECT - UNION

```
SELECT Country,
       LEFT(PostalCode,4) AS Sales,
       'Minimum' As Explanation
FROM customers c
WHERE LEFT(RIGHT(LEFT(PostalCode,4),4),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),3),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),2),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),1),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(PostalCode,4) <=ALL(SELECT LEFT(PostalCode,4)
                               FROM customers c_inner
                               WHERE c.Country=c_inner.Country)
UNION
SELECT Country,
       LEFT(PostalCode,4) AS Sales,
       'Maximum' As Explanation
FROM customers c
WHERE LEFT(RIGHT(LEFT(PostalCode,4),4),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),3),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),2),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),1),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
```

```
      LEFT(PostalCode,4) >=ALL(SELECT LEFT(PostalCode,4)
                               FROM customers c_inner
                               WHERE c.Country=c_inner.Country)
ORDER BY 1 ASC, 3 DESC
```

Country	Sales	Explanation
Austria	8010	Minimum
Austria	8010	Maximum
Denmark	1734	Minimum
Denmark	8200	Maximum
France	7501	Minimum
France	7501	Maximum
Germany	1220	Minimum
Germany	1477	Maximum
Italy	2410	Minimum
Italy	2410	Maximum
Mexico	0502	Minimum
Mexico	0502	Maximum
Spain	0802	Minimum
Spain	2803	Maximum
USA	8711	Minimum
USA	9720	Maximum

<=ALL und >=ALL in einer Anfrage

```
SELECT Country, LEFT(PostalCode,4) AS SalesMinMaxAvgCoincide
FROM customers c
WHERE LEFT(RIGHT(LEFT(PostalCode,4),4),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),3),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),2),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),1),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(PostalCode,4) <=ALL(SELECT LEFT(PostalCode,4)
                               FROM customers c_inner
                               WHERE c.Country=c_inner.Country)
      AND
      LEFT(PostalCode,4) >=ALL(SELECT LEFT(PostalCode,4)
                               FROM customers c_inner
                               WHERE c.Country=c_inner.Country)

Äquivalent (mit 'T=ALL(Q)' statt 'T<=ALL(Q) AND T>=ALL(Q)'):

SELECT Country, LEFT(PostalCode,4) AS SalesMinMaxAvgCoincide
FROM customers c
WHERE LEFT(RIGHT(LEFT(PostalCode,4),4),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),3),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),2),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),1),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(PostalCode,4) =ALL(SELECT LEFT(PostalCode,4)
                              FROM customers c_inner
                              WHERE c.Country=c_inner.Country)
```

Country	SalesMinMaxAvgCoincide
Mexico	0502
Austria	8010
Italy	2410
France	7501

SUM


```
SELECT Country, SUM(LEFT(PostalCode,4)) AS SumSales
FROM customers
WHERE LEFT(RIGHT(LEFT(PostalCode,4),4),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),3),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),2),1) IN
        ('0','1','2','3','4','5','6','7','8','9') AND
      LEFT(RIGHT(LEFT(PostalCode,4),1),1) IN
        ('0','1','2','3','4','5','6','7','8','9')
GROUP BY Country
ORDER BY 2 DESC
```

Country	SumSales
USA	18431
Denmark	9934
Austria	8010
France	7501
Spain	3605
Germany	2697
Italy	2410
Mexico	502

Beobachtung: MIN and MAX ermitteln Werte, die in der Tabelle selbst zu finden sind. COUNT, SUM und AVG berechnen Werte, die in der Regel nicht in der Tabelle selbst stehen.

Weitere Themen

- **NULL-Werte in SQL (Wahrheitstabellen für boolesche Operationen, Verhalten von Aggregationsfunktionen)**
- **SQL-Theoreme; z.B. 'IN ~ =ANY', 'NOT IN ~ <>ALL', 'NOT (op ALL) ~ opNeg ANY', 'NOT (op ANY) ~ opNeg ALL'**