# SQL für Fortgeschrittene

Das was in diesen Arbeitsblättern steht, ist für die Schüler gedacht, die gerne mehr über SQL lernen möchten. Es ist kein Teststoff, und wird im Unterricht auch nicht abgefragt.

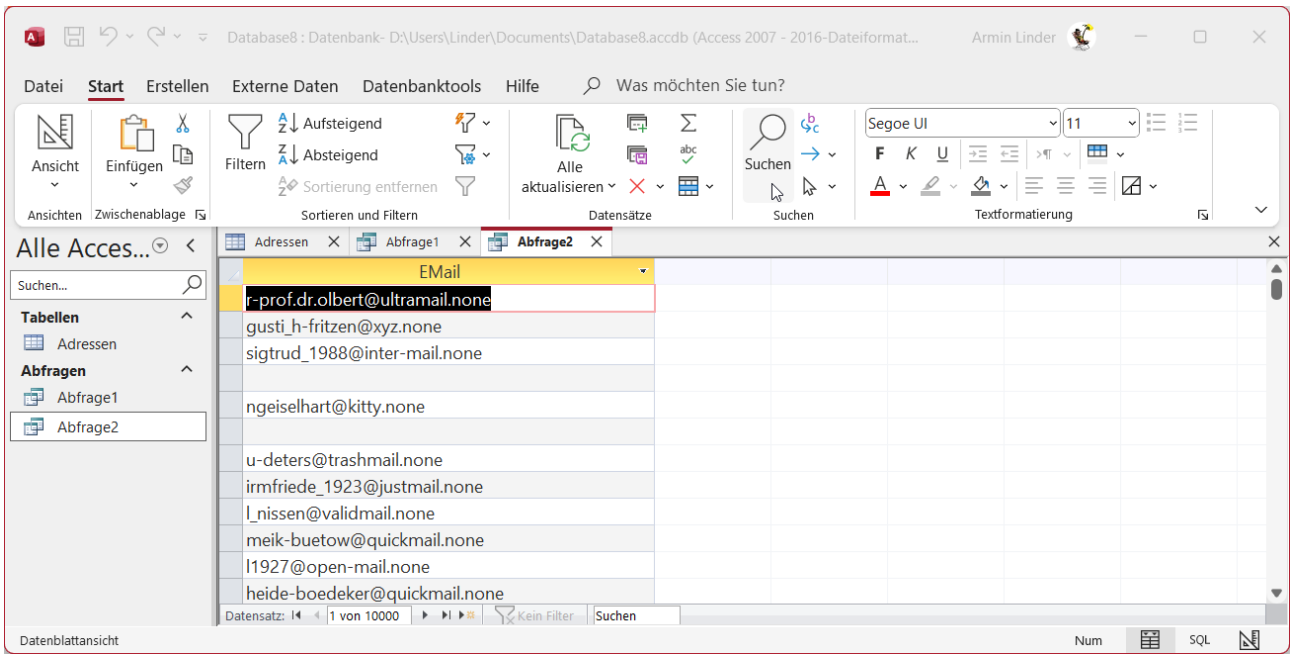
Wenn du Spaß an SQL hast, kannst du hier coole Abfragen lernen, die sogar richtige Programmierer nicht ohne Weiteres schreiben könnten.

Statt einfach die Lösung hinzuschreiben, möchte ich auch zeigen, wie man auf so etwas kommt. Ich gehe also Schritt für Schritt vor.

## Zählen aller Datensätze mit doppelten E-Mail Adressen

Erst einmal holen wir uns mit einer einfachen SELECT Anweisung alle E-Mail Adressen aus der Datenbank.

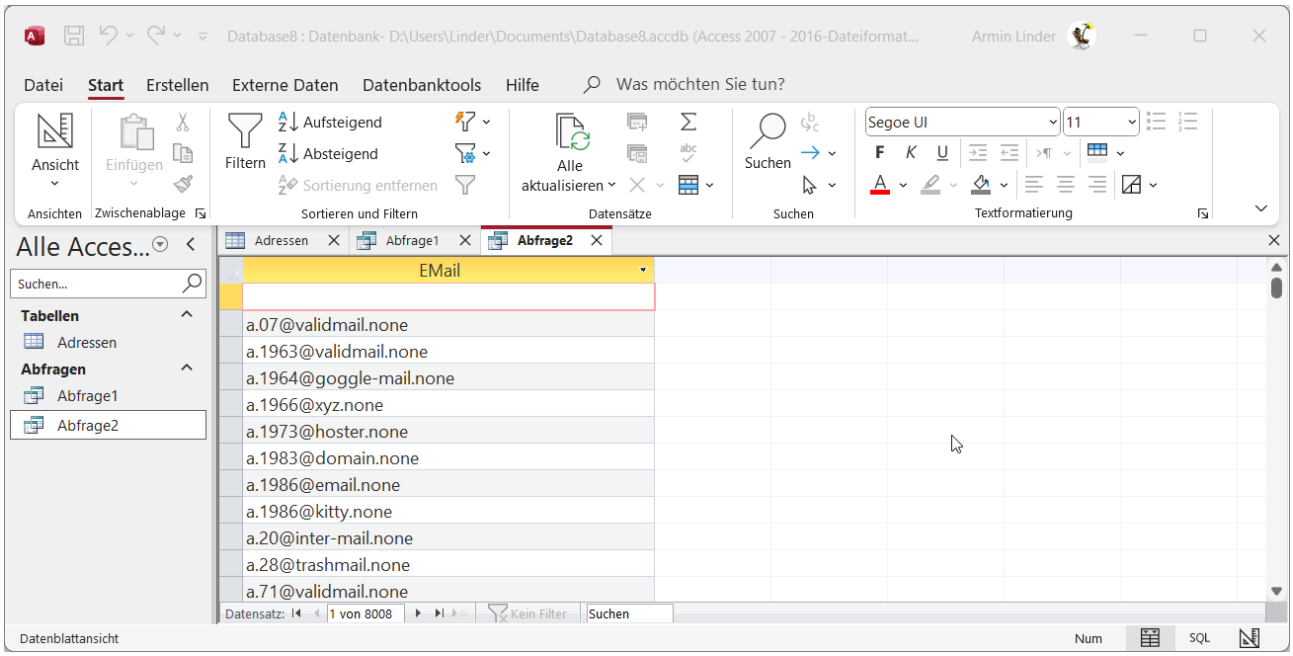
```
Select Email from Adressen
```




Na toll. 10.000 Zeilen. Und jetzt sortieren wir die Liste, und dann suchen wir uns alle die doppelt sind, oder was? Vergiss es. Es sind zu viele.

Da könnte uns jetzt SQL helfen. Es gibt da nämlich einen "Group by" Befehl, da fasst Access alle doppelten Datensätze zusammen. Doppelte werden weggelassen. Gedanke: wir lassen mal einfach die Doppelten weg, und schauen, welche übrigbleiben. Die sind nicht doppelt. und das was dann fehlt, war offenbar doppelt.

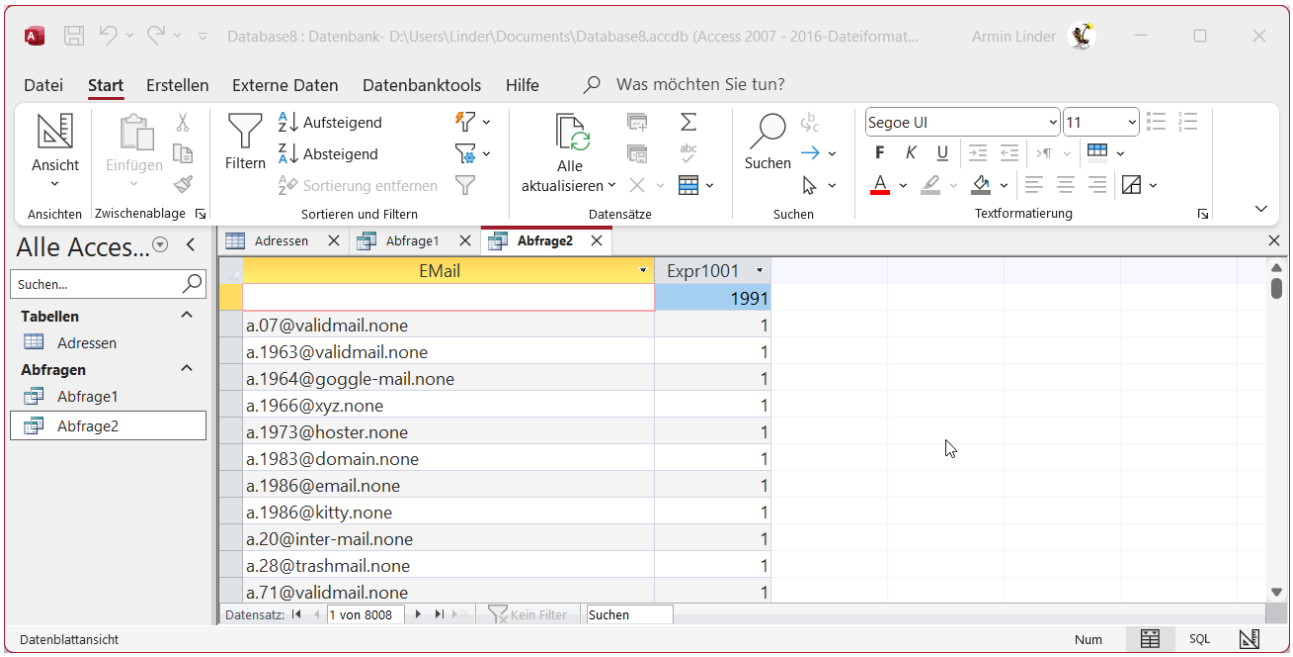
```
SELECT EMail from Adressen Group by EMail
```




Na toll. Jetzt sind es noch 8008 Datensätze. Bei denen die fehlen, war wohl die E-Mail Adresse doppelt. Aber was nützt mich das? Und so nebenbei kommt schon eine Warnklingel: es fehlen fast 1992 Datensätze. So viele doppelte E-Mail Adressen in meiner Datenbank? Das wäre nicht gut. Ist das überhaupt möglich? Mal sehen.

Ich möchte schon wissen, welche Datensätze *genau* da betroffen sind. Vielleicht kann ich dann sehen, wo die Doppelten herkommen? Also zurück an die Abfrage, wir müssen sie schlauer machen.

```
SELECT EMail,Count(*) from Adressen Group by EMail
```




Ookey. Also ein bisschen bringt uns das weiter. Count(*) ist eine SQL Funktion, die Datensätze zählen kann. Und "Group by" kennen wir schon, der Befehl fasst doppelte Felder zusammen.

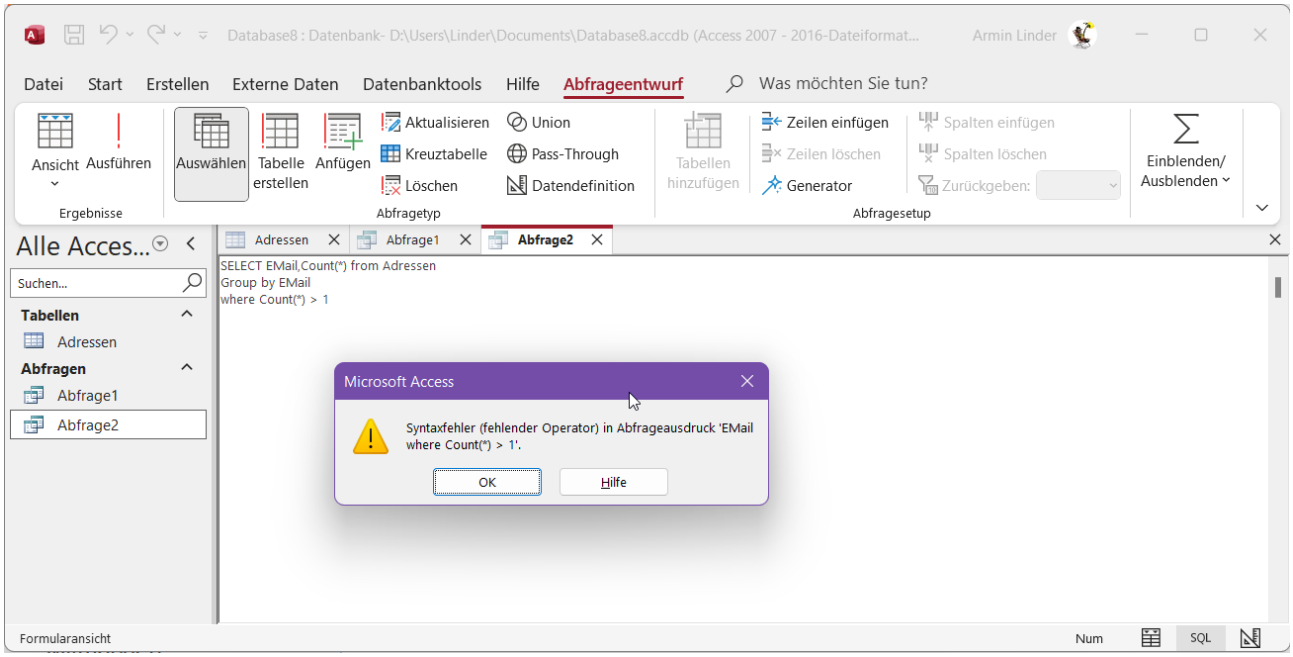
Beide zusammen machen also folgendes: doppelte Felder werden zusammengefasst, und count sagt, wie viele es waren. Bei einem Feld das überhaupt nicht doppelt ist, ist count gleich 1, eins mit einem Duplikat hat eine 2, gibt es eins sogar drei Mal, wäre es 3. usw.

Jetzt müssen wir in der Datenbank "nur" noch die Zeilen finden, in denen keine 1 steht. Eine haben wir zufällig ganz oben: da war die E-Mail Adresse gar nicht eingetragen. Die E-Mail Adresse ist also leer. Group by hat alle mit leerer E-Mail Adresse zusammengefasst, und count hat sie gezählt: es sind 1991. Aha, da kommt also die auffällig große Zahl her: bei vielen Datensätzen fehlt die E-Mail Adresse. Access zählt die als doppelt.

Es fehlen aber noch einige, da ist die E-Mail Adresse tatsächlich doppelt. Wie kriegen wir nun genau die heraus? Nun, eine Möglichkeit ist, sich an den SQL Befehl "WHERE" zu erinnern. Können wir den nicht zusammen mit Count verwenden, also so:

```
SELECT EMail,Count(*) from Adressen
Group by EMail
where Count(*) > 1
```

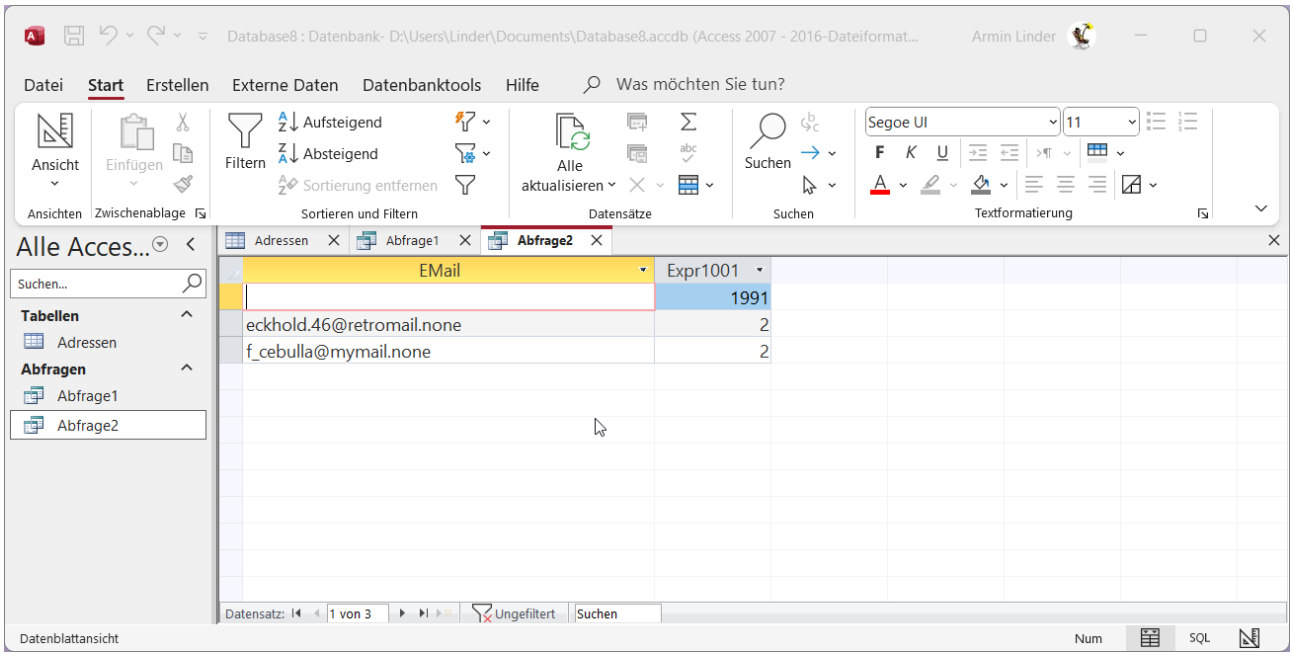
Die Idee war nicht schlecht, aber SQL macht Zicken.




Was Access eigentlich sagen sollte: man kann zusammen mit "WHERE" nur "richtige" Datenfelder verwenden, also solche, die auch in der Datenbank vorhanden sind. Count ist nicht Teil der Datenbank, sondern wird erst aus Datensätzen berechnet.

Um auf Count und andere Funktionen filtern zu können, muss man den SQL Befehl "HAVING" kennen. Der funktioniert ähnlich wie "WHERE", nur dass er statt einem Datenbankfeld einen Befehl erwartet, auf deren Ergebnis er dann filtert. COUNT ist so ein Befehl.

```
SELECT EMail,Count(*) from Adressen
Group by EMail
having count(*) > 1
```




Das ist schon besser jetzt. Nur … welche Datensätze sind das nun konkret? Ich möchte jetzt gerne alle Felder dieser Datensätze sehen, um z.B. die Betroffenen anzurufen, oder um ihnen einen Brief zu schreiben.

Das SQL um das herauszufinden gehört schon in die Profi-Liga:

```
SELECT *
FROM Adressen
INNER JOIN (SELECT nachname,vorname,email, COUNT(*)
FROM Adressen
GROUP BY nachname,vorname, email
HAVING count(*) > 1 ) b
ON Adressen.email = b.email
ORDER BY Adressen.email
```

Das Ergebnis schaut ziemlich beeindruckend aus, finde ich.


Aber wie funktioniert das? Diese komplizierte Abfrage musst du von innen nach außen lesen. Sie filtert zuerst – genauso wie oben gezeigt – mit den Befehlen ab „INNER JOIN“ alle Datensätze die doppelt sind. Vorher stand das Ergebnis auf dem Bildschirm, aber jetzt brauche sie jetzt nochmal, um auf die ursprüngliche Adresstabelle zuzugreifen, um dort die restlichen Felder zu holen. Das einsame „b“ hinter der letzten Klammer speichert die Ergebnistabelle der Duplikatssuche unter dem Namen „b“, damit ich weiter auf sie zugreifen kann.

INNER JOIN sucht in zwei Tabellen nach allen Datensätzen, die in beiden Tabellen enthalten sind, ON sagt dem Computer, welches Datenfeld er dabei vergleichen soll. Die eine Tabelle ist unsere Ausgangstabelle „Adressen“, die andere unsere temporäre Tabelle „b“ mit dem Ergebnis der Duplikatssuche,

Und da er aus „Adressen“ mit SELECT * alle Datenfelder geholt hat, bekomme ich nun alle angezeigt, sofern die E-Mail Adresse in der Tabelle „b“ enthalten war. Fertig.