# Modul 06: JOINS: LÖSUNGEN

*Systemvoraussetzungen: eine beliebige Version von SQL-Server und SSMS (Microsoft SQL Server Management Studio)*

*Tools: SQL-Server; SSMS (Microsoft SQL Server Management Studio)*

*Autor: Leonard Hlavin, Letzte Änderung: 19.04.2021*

## 1. Lösung zu Übung: Einfache JOINS verwenden

SELECT o.OrderID AS Bestellnummer

, o.CustomerID AS Kundennummer

, c.CompanyName AS Firmenname

, o.Freight AS Frachtkosten

FROM Customers AS c INNER JOIN Orders AS o ON c.CustomerID = o.CustomerID

WHERE c.Country = 'Brazil'

Hinweise:

* Da der Name CustomerID in zwei Tabellen vorkommt, müssen wir dazusagen, aus welcher Tabelle wir diese Information beziehen wollen; bei den anderen ist es optional, *sollte* aber immer dabeistehen.
* In diesem Fall können wir die CustomerID entweder aus der Customers- oder der Orders-Tabelle abfragen; da die Daten hier übereinstimmen, spielt es in diesem konkreten Fall keine Rolle, für welche wir uns entscheiden.
* Damit wir uns Tipparbeit ersparen, dürfen wir die Tabellennamen innerhalb der Abfrage mit einem Kürzel abkürzen (in diesem Fall c für die Customers-Tabelle und o für die Orders-Tabelle).
* Das Kürzel für die Tabellennamen gilt nur innerhalb dieser Abfrage und wird wie ein Spaltenalias mit „AS“ vergeben.
* Das „AS“ dürfen wir auch weglassen. Bei den Spaltenalias sollten wir das „AS“ zwecks besserer Lesbarkeit verwenden, obwohl es weggelassen werden darf, bei den Tabellen kann es weggelassen werden, ohne die Lesbarkeit zu beeinträchtigen:

FROM Customers c INNER JOIN Orders o ON c.CustomerID = o.CustomerID

## 2. Lösung zu Übung: JOINS und Wildcards verwenden

SELECT p.ProductName

, s.CompanyName

, s.ContactName

, s.Phone

FROM Products p INNER JOIN Suppliers s ON p.SupplierID = s.SupplierID

WHERE p.ProductName LIKE '%sauce%' OR p.ProductName LIKE N'%soße'

Hinweise:

* Wir brauchen einen JOIN der Tabellen Products (um den Produktnamen abfragen zu können) und Suppliers (um die Kontaktinformationen abfragen zu können).
* Wir verwenden einen INNER JOIN, denn wenn wir nachbestellen wollen, brauchen wir nur die Supplier, die uns schon Soßen geliefert haben.
* Wir verwenden ein LIKE und Wildcards, um mehrere Möglichkeiten abzudecken (also alle Produkte, bei denen im Namen „Sauce“ vorkommt – dann bekommen wir 2 Ergebnisse).
* Wollen wir auch die Produkte abdecken, bei denen im Namen „Soße“ vorkommt, bekommen wir ein Ergebnis mehr.

## 3. Lösung zu Übung: JOINS, WHERE und Serverfunktionen verwenden

SELECT o.OrderID

, CONCAT(e.FirstName, ' ', e.LastName) AS FullName

FROM Orders o INNER JOIN Employees e ON o.EmployeeID = e.EmployeeID

WHERE OrderID IN(10251, 10280, 10990, 11000)

Hinweise:

* Mit CONCAT können wir Vor- und Nachname in einer Spalte als FullName ausgeben.
* Wir verwenden einen INNER JOIN, denn wir brauchen nur solche Angestellte, die schon Bestellungen bearbeitet haben.

## 4. Lösung zu Übung: JOINS über mehrere Tabellen verwenden

SELECT o.OrderID

, o.CustomerID

, c.CompanyName

, p.ProductName

, od.Quantity

FROM Customers c INNER JOIN Orders o ON c.CustomerID = o.CustomerID

INNER JOIN [Order Details] od ON od.OrderID = o.OrderID

INNER JOIN Products p on p.ProductID = od.ProductID

WHERE p.ProductName LIKE '%chai%'

Hinweise:

* Wir verwenden INNER JOINS, denn wir brauchen nur die Kunden, die schon etwas bestellt haben (nämlich Chai Tee), und nur die Produkte, die schon bestellt wurden (wieder Chai Tee).
* Beim Verwenden von mehreren JOINS kommen die weiteren Tabellen zu den bereits verknüpften hinzu. Wenn es jemandem hilft, die Lesbarkeit zu verbessern, dürften die JOINS auch mit Klammern versehen werden:

FROM ((Customers c INNER JOIN Orders o ON c.CustomerID = o.CustomerID)

INNER JOIN [Order Details] od ON od.OrderID = o.OrderID)

INNER JOIN Products p on p.ProductID = od.ProductID

* Die JOINS untereinander zu schreiben und einzurücken hat keine Auswirkung auf die Funktionalität, sondern dient lediglich der besseren Lesbarkeit und Übersichtlichkeit.
* In diesem Fall gibt es nur ein Produkt namens „Chai“. Mit '%chai%' hätten wir aber auch andere Varianten von Chai-Tee abgedeckt, etwa, wenn es einen „Vanilla Chai“ oder einen „Premium Chai Tee“ gäbe.
* Vorsicht mit Zeichenfolgen und Wildcards: Gäbe es ein anderes Produkt, das die Zeichenfolge „chai“ enthält, ohne etwas mit Tee zu tun zu haben, so würde auch dieses ausgegeben werden.

## 5. Lösung zu Übung: JOINS über mehrere Tabellen, WHERE und Wildcards

SELECT o.OrderID

, c.CustomerID

, c.CompanyName

, p.ProductName

, od.Quantity

FROM Customers c INNER JOIN Orders o ON c.CustomerID = o.CustomerID

INNER JOIN [Order Details] od ON od.OrderID = o.OrderID

INNER JOIN Products p ON p.ProductID = od.ProductID

WHERE ProductName LIKE '%bier%'

OR ProductName LIKE '%lager%'

OR ProductName LIKE '%ale'

ORDER BY Quantity DESC, CompanyName -- ASC

Hinweise:

* JOINS wie in Übung 5.4
* Achtung auf die Aufgabenstellung: Wenn Sie statt '%ale' (soll mit „ale“ enden) '%ale%' (soll „ale“ enthalten) abgefragt haben, bekommen Sie auch Ergebnisse zurück, die mit Bier nichts zu tun haben.
* Im ORDER BY geben wir an, wonach die Ausgabe geordnet werden soll. Der Default ist ASC (ascending); wenn wir möchten, dass die Ausgabe alphabetisch oder vom kleinsten zum größten Wert geordnet wird, müssen wir keinen Zusatz zum Spaltennamen hinzufügen.
* Wenn wir vom größten zum kleinsten Wert oder in umgekehrter alphabetischer Reihenfolge ordnen wollen, verwenden wir den Zusatz DESC (descending, absteigend).

## 6. Lösung zu Übung: SELFJOIN

SELECT CONCAT(e1.FirstName, ' ', e1.LastName) AS Mitarbeiter

, e1.EmployeeID AS Mitarbeiternummer

, CONCAT(e2.FirstName, ' ', e2.LastName) AS Vorgesetzter

, e2.EmployeeID AS ChefID -- oder e1.ReportsTo

FROM Employees e1 LEFT JOIN Employees e2 ON e1.ReportsTo = e2.EmployeeID

Hinweise:

* Mit CONCAT können wir Vor- und Nachname in einer Spalte ausgeben.
* Welches ALIAS wir für die Tabellen vergeben, ist egal, solange wir selbst uns damit auskennen. Wir könnten auch die erste Tabelle z.B. „emp“ und die hypothetische zweite Tabelle „boss“ nennen:

FROM Employees emp LEFT JOIN Employees boss ON emp.ReportsTo = boss.EmployeeID

* Wenn auch die Employees, die keinen Vorgesetzten haben (entweder, weil keiner zugewiesen ist oder, so wie hier, weil sie selbst der Chef sind) ausgegeben werden sollen, verwenden wir einen OUTER JOIN, hier einen LEFT JOIN.
* Wenn der Chef selbst oder solche Angestellte, die keinen Vorgesetzten haben (Teamleiter, noch nicht zugeordnet, …) NICHT mit ausgegeben werden sollen, verwenden wir einen INNER JOIN.
* Beachten Sie: wenn wir ein CONCAT verwenden, steht in der Spalte „Vorgesetzter“ beim Chef nicht „NULL“ wie in der ID-Spalte daneben, denn in der Spalte steht jetzt das Leerzeichen aus dem CONCAT!

## 7. Lösung zu Übung: SELFJOIN

SELECT a.CompanyName AS Kunde

, a.City AS Stadt

, b.CompanyName AS Mitfahrgelegenheit

, b.City AS Kontrollfeld

FROM Customers a INNER JOIN Customers b ON a.City = b.City

WHERE a.CustomerID != b.CustomerID

ORDER BY a.City, a.CompanyName

Hinweise:

* Wir erstellen den Selfjoin auf der Spalte City, denn wir wollen ja die herausfinden, die in der hypothetischen zweiten Tabelle in derselben Stadt wohnen, wie die in der eigentlichen Customers-Tabelle.
* Wir verwenden einen INNER JOIN; hier haben wir keine, wo es keine Übereinstimmungen gibt.
* Mit WHERE a.CustomerID != b.CustomerID schließen wir aus, dass z.B. „ALFKI“ neben „ALFKI“ (also sich selbst) steht.