# Modul 04: WHERE-Klausel: 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: Abfragen von einfachen Bedingungen

SELECT CustomerID

, CompanyName

, ContactName

, Phone

FROM Customers

WHERE Country = 'France'

Hinweis: Die genauen Spalten sind für unsere Übung im Moment nicht relevant; sollten Sie mehr oder weniger als oben gewählt haben, ist das auch in Ordnung.

## 2. Lösung zu Übung: Abfragen von mehreren Bedingungen

SELECT CustomerID

, CompanyName

, ContactName

, Phone

FROM Customers

WHERE Country = 'Argentina' AND City = 'Buenos Aires'

Hinweis:

Ganz ähnlich wie bei Übung 3.1, allerdings müssen wir hier ein AND verwenden. Beide Bedingungen (Argentina und Buenos Aires) *müssen* zutreffen. *Gäbe* es ein „Buenos Aires“, das nicht in Argentinien liegt, dann würde das von dieser Abfrage nicht erfasst werden.

## 3. Lösung zu Übung: Abfragen von mehreren Bedingungen

SELECT CustomerID

, CompanyName

, ContactName

, Phone

FROM Customers

WHERE Country = 'Spain' OR Country = 'Portugal'

Oder für Fortgeschrittene:

SELECT CustomerID

, CompanyName

, ContactName

, Phone

FROM Customers

WHERE Country IN('Spain', 'Portugal')

Hinweise:

* Ganz ähnlich wie bei den Übungen 3.1 und 3.2, allerdings brauchen wir hier ein OR! Würden wir hier ein AND verwenden, müsste der Kunde zugleich in Portugal und Spanien ansässig sein.
* Fragen wir im WHERE Bedingungen für dieselbe Spalte mehrfach ab (soll also entweder A oder B oder C… zutreffen), dann können wir das auch mit einem **IN**('A', 'B', 'C') abfragen.

## 4. Lösung zu Übung: Einfache Bedingungen abfragen

SELECT \*

FROM Products

WHERE UnitsInStock > 100

Hinweis: SELECT \* wählt alle Spalten der Tabelle aus. Das sollte in der Realität aber nicht verwendet werden; es könnte sich z.B. etwas an der Tabelle ändern, und dann funktioniert die Abfrage nicht mehr.

Auch wenn wir alle Spalten einer Tabelle ausgeben möchten, sollten sie einzeln angeführt werden. Strenggenommen wäre Folgendes korrekt:

SELECT ProductID

, ProductName

, SupplierID

, CategoryID

, QuantityPerUnit

, UnitPrice

, UnitsInStock

, UnitsOnOrder

, ReorderLevel

, Discontinued

FROM Products

WHERE UnitsInStock > 100

## 5. Lösung zu Übung: Between

SELECT ProductID

, ProductName

, UnitPrice

FROM Products

WHERE ProductID BETWEEN 10 AND 15

Hinweise:

* Bereiche (Wert liegt zwischen x und y) können wir auch mit BETWEEN abdecken.
* BETWEEN macht ein >= und <=, d.h. die angegebenen Grenzwerte sind noch enthalten.
* Wir könnten diese Übung auch mit >= und <= lösen, auch das wäre korrekt:

SELECT ProductID

, ProductName

, UnitPrice

FROM Products

WHERE ProductID >= 10 AND ProductID <= 15

## 6. Lösung zu Übung: IN()

SELECT ProductID

, ProductName

, UnitPrice

FROM Products

WHERE SupplierID IN(2,7,15)

Hinweise:

* Fragen wir im WHERE Bedingungen für dieselbe Spalte mehrfach ab (soll also entweder A oder B oder C… zutreffen), dann können wir das auch mit einem **IN**('A', 'B', 'C') abfragen.
* Da es sich bei der SupplierID um den Datentyp int handelt, und nicht um Text, setzen wir 2, 7 und 15 NICHT unter Hochkommata.
* Diese Abfrage könnte auch mit OR gelöst werden, dann haben wir aber ein bisschen mehr Tipparbeit:

SELECT ProductID

, ProductName

, UnitPrice

FROM Products

WHERE SupplierID = 2 OR SupplierID = 7 OR SupplierID = 15

## 7. Lösung zu Übung: Abfragen von mehreren Bedingungen

SELECT ProductID

, ProductName

, UnitPrice

, UnitsInStock

FROM Products

WHERE SupplierID IN(5, 10, 15) AND UnitsInStock > 10 AND UnitPrice < 100

Hinweise:

* Wir können mehrere Bedingungen aus mehreren, nicht unbedingt zusammengehörigen Spalten in einem WHERE abfragen. Werden diese mit einem AND verknüpft, müssen alle Bedingungen zutreffen.
* Die SupplierIDs 5, 10 und 15 können wir mit einem IN() zusammenfassen.

## 8. Lösung zu Übung: WHERE und NULL; Datumsfunktionen

SELECT OrderID

, FORMAT(ShippedDate, 'd', 'de-de') AS Lieferdatum

, FORMAT(RequiredDate, 'd', 'de-de') AS Wunschtermin

, DATEDIFF(dd, RequiredDate, ShippedDate) AS Lieferverzögerung

FROM Orders

WHERE DATEDIFF(dd, RequiredDate, ShippedDate) IS NOT NULL

-- statt FORMAT können wir auch CONVERT verwenden, um das Datum in eine leserliche Form zu bringen:

SELECT OrderID

, CONVERT(varchar, ShippedDate, 104) AS Lieferdatum

, CONVERT(varchar, RequiredDate, 104) AS Wunschtermin

, DATEDIFF(dd, RequiredDate, ShippedDate) AS Lieferverzögerung

FROM Orders

WHERE DATEDIFF(dd, RequiredDate, ShippedDate) IS NOT NULL

Hinweise:

* FORMAT verfügt über einen sogenannten Culture-Parameter. Den kennen wir auch von Webseiten, die in mehreren Sprachen verfügbar sind („de-de“, „en-gb“ usw.). Die Liste der Culture-Parameter gibt es in der Microsoft-Dokumentation: <https://docs.microsoft.com/de-de/bingmaps/rest-services/common-parameters-and-types/supported-culture-codes>
* CONVERT hingegen verwendet einen sogenannten Style-Parameter; hier geben wir über einen Zahlencode ein, in welcher Form die Ausgabe des Datums erfolgen soll. Eine Übersicht über die möglichen Style-Parameter gibt es in der Microsoft-Dokumentation: <https://docs.microsoft.com/en-us/sql/t-sql/functions/cast-and-convert-transact-sql?view=sql-server-2017#date-and-time-styles>

In DATEDIFF verwenden wir in den allermeisten Fällen das ältere Datum als Startdatum und das jüngere/neuere Datum als Enddatum. Manchmal macht die Fragestellung aber einen großen Unterschied. In diesem Beispiel war die *Lieferverzögerung* gefragt. Hätte die Fragestellung aber gelautet: *„Geben Sie an, wie viele Tage noch Zeit sind, um fristgerecht zu liefern“,* hätten wir Start- und Enddatum in der Abfrage genau umkehren müssen!