



FUTURE **LOGISTICS** [AGINE HERE]

Name: Hossein Akbari
Matrikelnummer: 39940
Hochschulmail: hosakbari@mail.hs-bremerhaven.de
Dozent: Prof. Dr. Nadja Petram
Zeitraum der Bearbeitung: 24.10.2024
Datum der Fertigstellung: 12.02.2025

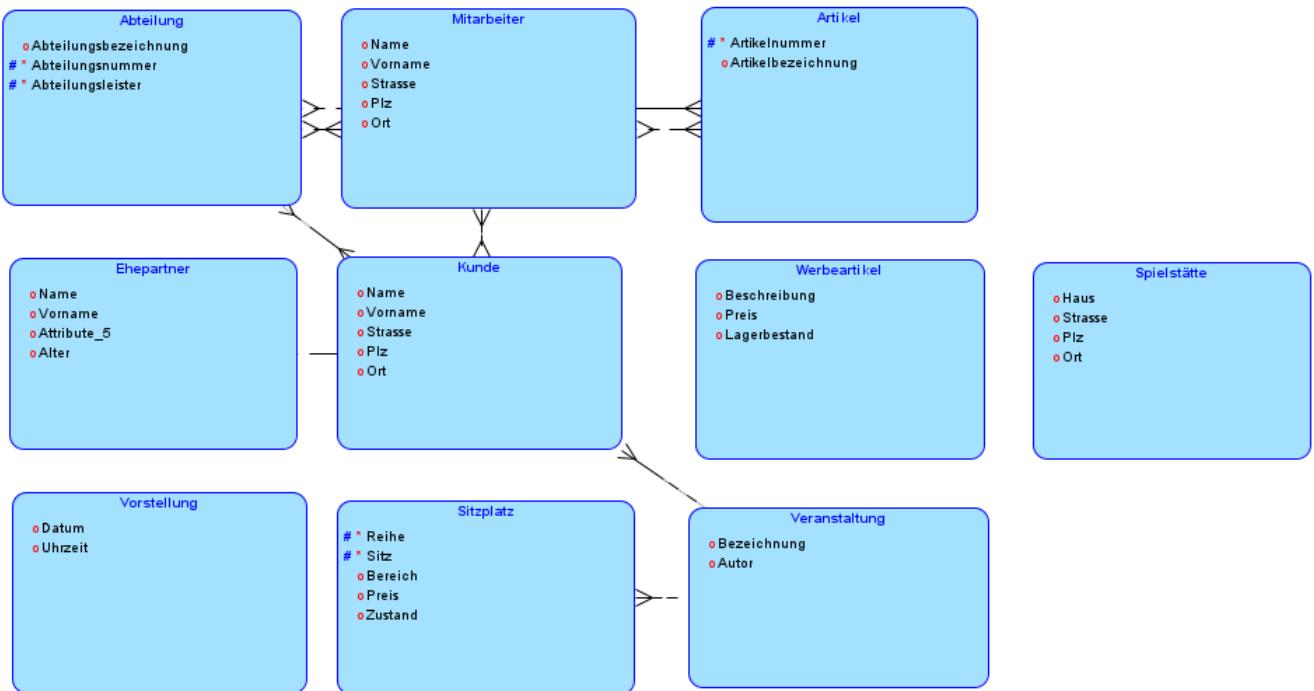
Inhaltsverzeichnis

Semesteraufgabe 17.10.2024	2
Semesteraufgabe 24.10.2024	4
FutureLogistics – Die Logistik der Zukunft im Jahr 2044.....	4
Beschreibung.....	4
Ist-Analyse.....	4
Problemanalyse.....	4
Visualisierung	5
.....	6
Objekttypen und Beziehungen	7
Entity-Entwurf.....	7
Erstellen Sie ein ER-Diagramm	8
Bilden Sie das ER-Diagramm in ein relationales Schema ab	9
Erläutern Sie die Fremdschlüssel-Beziehung	9
Stellen Sie die einzelnen Relationen/Tabellen in tabellarischer Form und als CREATE-TABLE-Anweisungen dar. Erzeugen Sie alle Tabellen mit SQL-Befehlen. (Nur CREATE-TABLE-Anweisung verwenden, ohne ALTER TABLE, Index etc...).....	11
Füllen Sie die Datenbank mit einigen Testdaten, verwenden Sie dabei SQL-Befehle (mindestens drei Insert- Operationen pro Tabelle).....	13
Führen Sie folgende SQL Operationen: SELECT * FROM TABELLE; SELECT SPALTE1, SPALTE2,... FROM TABELLE; SELECT SPALTE1, SPALTE2,... FROM TABELLE ORDER BY SPALTE1, SPALTE2,...; SELECT SPALTE1, SPALTE2,... FROM TABELLE ORDER BY SPALTE1,... DESC; SELECT * FROM TABELLE WHERE BEDINGUNG; SELECT DISTINCT SPALTE1,... FROM TABELLE	18
Führen Sie folgende SQL-Operationen (4 Beispiele pro Operation) mit Ihren Testdaten aus: UPDATE, ALTER, DELETE; Relationen mit Nullmarken (4 Beispiele); Problematik „Logistik in 20 Jahren“	36
Aggregatfunktionen: COUNT(*), Anzahl der Tupel; COUNT(attr), Anzahl der Tupel ohne Null für attr; COUNT(DISTINCT attr), Anzahl der verschiedenen Tupel ohne Null für attr; SUM(attr), Summe der Werte von attr (nur für numerische Attribute); AVG(attr), Durchschnitt der Werte von attr (nur für numerische Attribute), MIN(attr), MAX(attr); Kombination von GROUP BY, ORDER BY; HAVING, Aggregatfunktionen; Problematik „Logistik in 20 Jahren“	72
Problematik „Logistik in 20 Jahren“; Führen Sie folgende SQL-Operationen (2 Beispiele pro Operation) mit Ihren Testdaten aus: Mengen-Operationen (OR), Mengen-Operationen – AND, Der natürliche Verbund (Join), INNER JOIN, Kartesisches Produkt (CROSS – JOIN), AS, Linker äußerer (Left Outer) Join, Rechter äußerer (Right Outer) Join, Right Join.....	86
Normalisieren Sie Relationen bis zur 3. Normalform.	97
Aufgabe 16.01.2025: - Erzeugen Sie 5 virtuelle Views (Datensichten); - Schreiben Sie die formulierten Unterabfragen mit Hilfe von SQL; - Anschließend einen Screenshot hinzufügen; - Problematik „Logistik in 20 Jahren“	125
Quellenverzeichnis.....	133

Semesteraufgabe 17.10.2024

Im Rahmen der Semesteraufgabe 17.10.2024 sollten wir die Software „SQL Developer Data Modeler“ installieren. Im Nachhinein sollten wir, um mit dem Umgang der Software vertraut zu werden, aus der Vorlesung „Datenbanken 1: Einführung“, Folie 51:

„Attribute und Schlüssel“, die Objekttypen, sowie die Attribute davon, in der Software einmal modellieren: Des Weiteren sollten einige Beziehungenarten (nicht alle) zwischen diesen Objekttypen erstellt werden:



Semesteraufgabe 24.10.2024

Im Rahmen der Semesteraufgabe 24.10.2024 soll eine Beschreibung und Visualisierung des Projekts „Logistik in 20 Jahren“ erfolgen. Hierzu soll für das Thema „Logistik in 20 Jahren“ eine Vision dessen entworfen werden, wie die Logistik wie wir sie heute kennen, durch technologische Innovationen und Nachhaltigkeitsbestrebungen weiterentwickelt werden könnte. Als Projekt für die Realisierung dieser Vision wurde das folgende Thema ausgedacht:

FutureLogistics – Die Logistik der Zukunft im Jahr 2044

Beschreibung

FutureLogistics ist ein visionäres Projekt für die Zukunft, mit der der Wandel der Logistikindustrie in den kommenden 20 Jahren beschrieben wird. Mit diesem Projekt soll das Ziel erreicht werden, eine nachhaltige und hochgradig effiziente Logistiklösung zu entwickeln, basierend auf die modernsten Technologien, mit der die Anforderungen einer globalisierten und digitalisierten Welt erfüllt wird. Hauptfokus des Projektes wird sein: Automatisierung, Künstliche Intelligenz, erneuerbare Energien und umweltfreundliche Praktiken, um die Logistikkette von der Produktion bis zur Auslieferung zu optimieren.

Ist-Analyse

Aktuell ist die Logistikbranche eine der wichtigsten, jedoch auch eine der ressourcenintensivsten Branchen der Weltwirtschaft. Die täglichen Abläufe in der Logistik basieren größtenteils auf traditionellen Methoden, die durch hohe Abhängigkeit von fossilen Brennstoffen und ineffizienten Prozessen gekennzeichnet sind. Der stetig steigende Treibstoffverbrauch sowie lange Transportwege führen zu erheblichen Umweltbelastungen und hohen Betriebskosten. Besonders in urbanen Gebieten belastet der Verkehr die Effizienz der Lieferketten, während in ländlichen Gebieten oft die nötigen Strukturen fehlen, um Lieferungen zeitnah abzuwickeln.

Ein weiterer zentraler Aspekt der aktuellen Situation in der Logistikbranche ist die menschliche Komponente. Logistische Abläufe, die stark auf manuelle Tätigkeiten angewiesen sind, bergen eine hohe Wahrscheinlichkeit für Fehler und Ineffizienzen. Traditionelle Prozesse, bei denen Daten oft manuell verarbeitet werden, erschweren es Unternehmen, einen ganzheitlichen Überblick zu behalten und Potenziale zur Optimierung zu erkennen. Manuelle Eingriffe verzögern häufig den Betrieb und erschweren die Möglichkeit einer vollständigen Prozessautomatisierung, die die Effizienz und Qualität der Abläufe steigern könnte.

Schließlich stehen Logistikunternehmen vor dem wachsenden Druck, sich an die steigenden Anforderungen des E-Commerce und die Erwartungen der Kunden anzupassen. Die Nachfrage nach schnellen und flexiblen Lieferungen hat sich in den letzten Jahren enorm gesteigert, wodurch die Branche gezwungen ist, neue Technologien wie Drohnen, autonome Fahrzeuge und intelligente Lagerverwaltungssysteme zu testen. Allerdings werden solche Technologien derzeit häufig nur in Pilotprojekten eingesetzt, da die umfassende Integration von Automatisierungslösungen hohe Investitionen und erweiterte Datenverarbeitungskapazitäten erfordert.

Problemanalyse

Die Logistikbranche ist heute mit einer Reihe von Herausforderungen konfrontiert, die ihre langfristige Entwicklung und Nachhaltigkeit gefährden. Einer der Hauptfaktoren ist die starke Abhängigkeit von fossilen Brennstoffen, was nicht nur die Kosten in die Höhe treibt, sondern auch erheblich zum globalen CO₂-Ausstoß beiträgt. Der Druck, umweltfreundlichere Alternativen zu finden, nimmt kontinuierlich zu, doch die Umstellung auf emissionsfreie Fahrzeuge oder grüne Energien ist aufgrund fehlender Infrastruktur und hoher Umstellungskosten schwierig. Gleichzeitig sind die hohen Emissionen ein wachsendes Problem für die Einhaltung globaler Klimaziele.

Neben den ökologischen Herausforderungen steht die Branche auch vor knappen Ressourcen und ineffizienten Transportwegen, die die Abläufe und die Wirtschaftlichkeit beeinträchtigen. Besonders in dicht besiedelten Gebieten sind die Straßen oft überlastet, was zu Verzögerungen und höherem Energieverbrauch führt. In weniger gut erschlossenen Regionen sind hingegen oft die logistischen Kapazitäten unzureichend, was die Flexibilität und Reaktionsfähigkeit der Unternehmen einschränken. Diese strukturellen Herausforderungen zeigen, wie notwendig eine umfassende Transformation hin zu einer effizienteren Logistik ist.

Darüber hinaus sind die Anforderungen an Datenschutz und Datensicherheit durch den zunehmenden Einsatz von Technologien wie vernetzten Geräten und Künstlicher Intelligenz gestiegen. Viele Unternehmen sammeln und

verarbeiten riesige Mengen an Kundendaten, was das Risiko von Datenschutzverletzungen erhöht und eine strenge Einhaltung der gesetzlichen Vorschriften erforderlich macht. Diese neuen Technologien stellen nicht nur regulatorische, sondern auch ethische Herausforderungen dar, die die Logistikunternehmen bewältigen müssen, um das Vertrauen ihrer Kunden aufrechtzuerhalten und ihre Wettbewerbsfähigkeit zu sichern.

Visualisierung

Außenansicht: Das FutureLogistics-Zentrum ist ein hochmodernes, autonom betriebenes Logistikzentrum, das sowohl ästhetisch als auch ökologisch durchdacht gestaltet ist. Auf dem Dach befinden sich großflächige Solarzellen, die das Gebäude und die Ladeinfrastruktur mit erneuerbarer Energie versorgen. Rund um das Zentrum sind zahlreiche Ladestationen für Elektrofahrzeuge angebracht, die es Lkws und Lieferfahrzeugen ermöglichen, während der Be- und Entladezeiten emissionsfrei Energie aufzutanken. Ein spezieller Drohnenlandeplatz befindet sich auf einer erhöhten Ebene des Gebäudes und ermöglicht eine schnelle, automatisierte Abwicklung von Lieferungen und Abholungen über die Luft – ideal für urbane Gebiete und schwer zugängliche Regionen.

Innenansicht: Das Innere des Logistikzentrums ist vollständig automatisiert und auf maximale Effizienz und Präzision ausgelegt. Intelligente Roboter transportieren und sortieren Waren in modularen Regalsystemen, während KI-gesteuerte Maschinen die Artikel für den Versand vorbereiten und verpacken. Roboterarme greifen gezielt nach Waren und kommissionieren diese je nach Zielort und Lieferpriorität. Alle Prozesse sind miteinander vernetzt und arbeiten ohne menschliche Intervention. Ein zentralisiertes Kontrollsysteem koordiniert die Roboterbewegungen in Echtzeit und stellt sicher, dass jede Lieferung pünktlich und fehlerfrei bearbeitet wird.

Technologische Visualisierung: FutureLogistics bietet eine hochmoderne Plattform zur Überwachung der gesamten Lieferkette in Echtzeit. Die Plattform ist für Kunden als App verfügbar, die eine vollständige Transparenz über den Status ihrer Lieferungen bietet. Kunden können jederzeit den Standort und den voraussichtlichen Lieferzeitpunkt einsehen und werden bei Verzögerungen sofort benachrichtigt. Die Plattform nutzt dabei KI-Algorithmen, die auf Basis der Echtzeit-Daten Optimierungen in der Routenplanung und Ressourcenzuweisung vornehmen. Die Benutzeroberfläche ist intuitiv gestaltet, sodass alle Statusinformationen, einschließlich der ökologischen Auswirkungen jeder Lieferung, übersichtlich präsentiert werden.





Objekttypen und Beziehungen

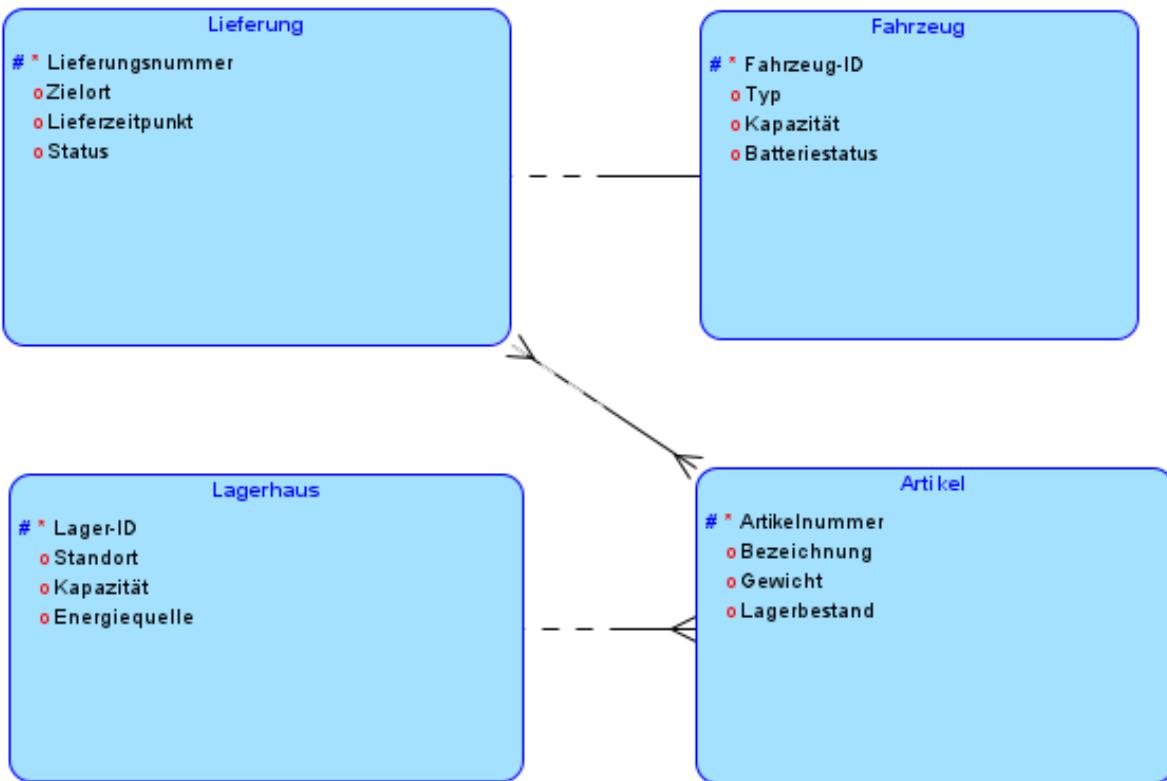
Des Weiteren sollten wir zu unserem obigen Beispiel 4 Objekttypen bzw. Objektklassen, mindestens 4 Attribute pro Klasse, mindestens 3 Beziehungen und Schlüssel definieren.

Objekttyp	Attribute	Primärschlüssel	Beschreibung
Lieferung	Lieferungsnummer, Zielort, Lieferzeitpunkt, Status	Lieferungsnummer	Eindeutige Identifizierung der Lieferung, Zielort, geplante Ankunft und Lieferstatus
Fahrzeug	Fahrzeug-ID, Typ, Kapazität, Batteriestatus	Fahrzeug-ID	Eindeutige Identifizierung des Fahrzeugs, Typ (z.B. Elektro-LKW, Drohne) und Kapazität
Lagerhaus	Lager-ID, Standort, Kapazität, Energiequelle	Lager-ID	Eindeutige Identifizierung des Lagerhauses, Standort, maximale Kapazität und Energiequelle
Artikel	Artikelnummer, Bezeichnung, Gewicht, Lagerbestand	Artikelnummer	Eindeutige Identifizierung des Artikels, Beschreibung, Gewicht und aktueller Bestand

Beziehungstyp	Objekttyp 1	Objekttyp 2	Beziehung	Beschreibung
1:1	Fahrzeug	Lieferung	Fahrzeug - Lieferung	Jede Lieferung wird genau einem Fahrzeug zugeordnet, und jedes Fahrzeug ist nur für eine Lieferung gleichzeitig im Einsatz
1:n	Lagerhaus	Artikel	Lagerhaus - Artikel	Ein Lagerhaus kann mehrere Artikel speichern, aber jeder Artikel befindet sich in einem bestimmten Lagerhaus
m:n	Lieferung	Artikel	Lieferung - Artikel	Eine Lieferung kann mehrere Artikel enthalten, und ein Artikel kann in mehreren Lieferungen vorkommen.

Entity-Entwurf

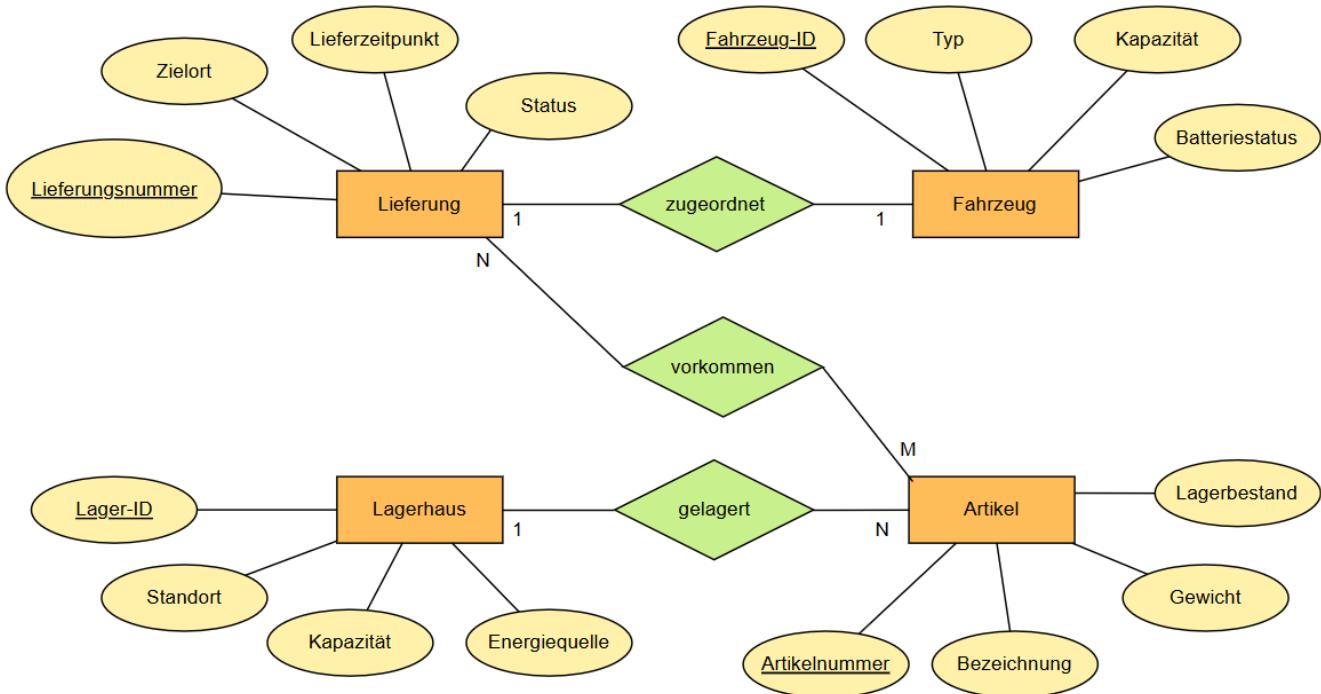
Weiterhin wurde von uns erwartet einen Entity-Entwurf durchzuführen



Der Entity-Entwurf beschreibt vier zentrale Entitäten: Lieferung, Fahrzeug, Lagerhaus und Artikel. Die Entität Lieferung umfasst Attribute wie Lieferungsnummer, Zielort, Lieferungszeitpunkt und Status und steht in einer 1:1-Beziehung mit Fahrzeug, welches ebenfalls Attribute wie Fahrzeug-ID, Typ, Kapazität und Batteriestatus besitzt. Lagerhaus mit den Attributen Lager-ID, Standort, Kapazität und Energiequelle eine 1:n-Beziehung zu Artikel, während Artikel mit Artikelnummer, Bezeichnung, Gewicht und Lagerbestand in einem n:m-Beziehung zu Lieferung steht. Diese Struktur ermöglicht eine effiziente Verwaltung von Lieferungen, Transportmitteln und Lagerbeständen und bildet die Basis für ein modernes Logistiksystem.

Erstellen Sie ein ER-Diagramm

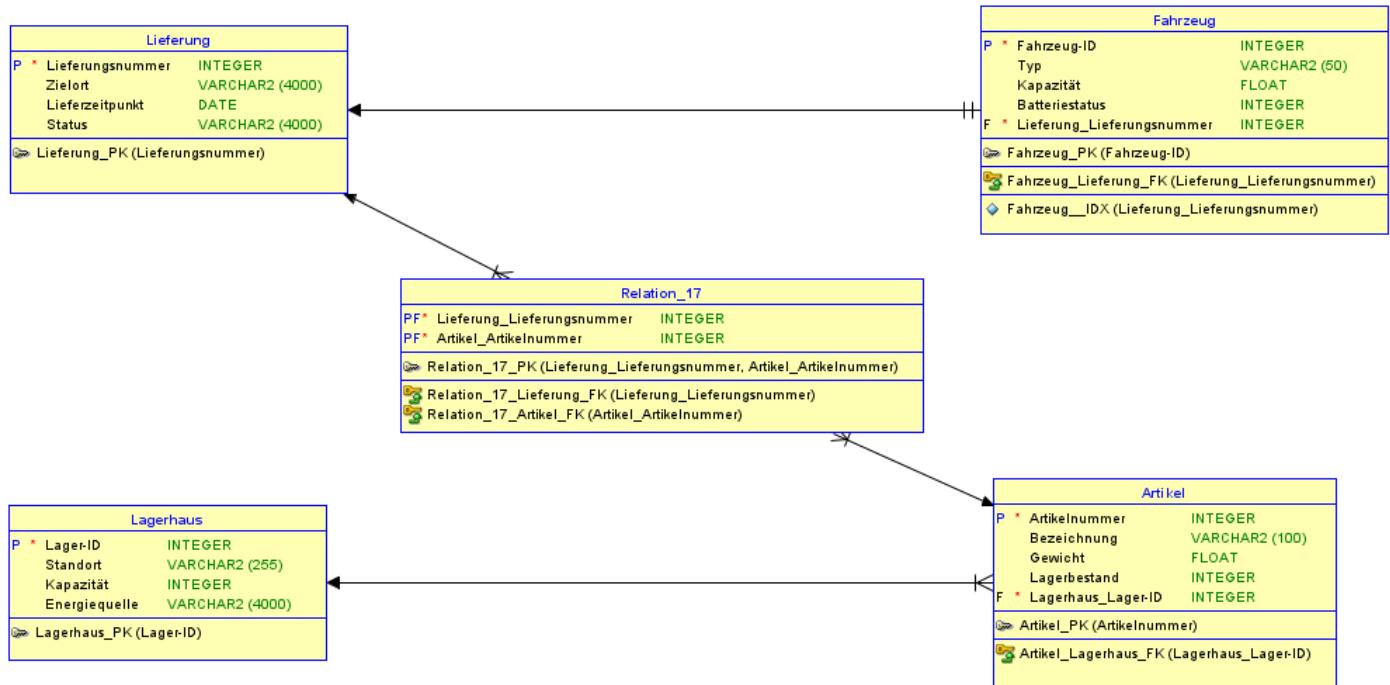
Uns wurde ebenfalls mitgeteilt ein Entity-Relationship-Diagramm zu erstellen.



Das Entity-Relationship-Diagramm für das Projekt FutureLogistics zeigt die zentrale Struktur und die Beziehungen zwischen den Entitäten Lieferung, Fahrzeug, Lagerhaus und Artikel in einem modernen Logistiksystem. Die Entität Lieferung umfasst Attribute wie Lieferungsnummer (Primärschlüssel), Zielort, Lieferzeitpunkt und Status, die zur eindeutigen Identifikation und Verwaltung jeder Lieferung dienen. Eine Lieferung wird in einer 1:1-Beziehung einem Fahrzeug zugeordnet, das mit Attributen wie Fahrzeug-ID, Typ, Kapazität und Batteriestatus definiert ist. Dadurch wird sichergestellt, dass jede Lieferung von einem spezifischen Fahrzeug ausgeführt wird. Gleichzeitig besteht eine m:n-Beziehung zwischen Lieferung und Artikel, sodass eine Lieferung mehrere Artikel enthalten kann, und umgekehrt kann ein Artikel in mehreren Lieferungen vorkommen. Die Entität Lagerhaus ist durch Attribute wie Lager-ID, Standort, Kapazität und Energiequelle charakterisiert und steht in einer 1:n-Beziehung zu Artikel, was bedeutet, dass ein Lagerhaus mehrere Artikel lagern kann, jeder Artikel jedoch nur in einem bestimmten Lagerhaus gespeichert ist. Schließlich beschreibt die Entität Artikel die Eigenschaften der Artikel mit Artikelnummer (Primärschlüssel), Bezeichnung, Gewicht und Lagerbestand. Diese Struktur ermöglicht eine effiziente Verwaltung von Lieferungen, Fahrzeugen, Lagerhäusern und Artikeln und bildet somit die Grundlage für ein optimiertes Logistiksystem, das flexibel auf verschiedene Anforderungen reagieren kann.

Bilden Sie das ER-Diagramm in ein relationales Schema ab

Im Nachhinein sollten wir das ER-Diagramm, welches wir davor erstellt haben, in ein relationales Schema abbilden.



Erläutern Sie die Fremdschlüssel-Beziehung

ER-Modell	Relationales Schema
Lieferung <ul style="list-style-type: none"> # * Lieferungsnummer o Zielort o Lieferzeitpunkt o Status 	Lieferung <ul style="list-style-type: none"> P * Lieferungsnummer: INTEGER Zielort: VARCHAR2 (4000) Lieferzeitpunkt: DATE Status: VARCHAR2 (4000) <p>Lieferung_PK (Lieferungsnummer)</p>
Beschreibung	
Lieferung: {Lieferungsnummer: INTEGER, Zielort: VARCHAR (4000), Lieferzeitpunkt: DATE, Status: VARCHAR2 (4000)}	

ER-Modell	Relationales Schema
Fahrzeug <ul style="list-style-type: none"> # * Fahrzeug-ID: INTEGER o Typ: VARCHAR (50) o Kapazität: FLOAT o Batteriestatus: INTEGER 	Fahrzeug <ul style="list-style-type: none"> P * Fahrzeug-ID: INTEGER Typ: VARCHAR2 (50) Kapazität: FLOAT Batteriestatus: INTEGER <p>F * Lieferung_Lieferungsnummer (Lieferung_Lieferungsnummer)</p> <p>Fahrzeug_PK (Fahrzeug-ID)</p> <p>Fahrzeug_Lieferung_FK (Lieferung_Lieferungsnummer)</p> <p>Fahrzeug_IDX (Lieferung_Lieferungsnummer)</p>
Beschreibung	

Fahrzeug: {Fahrzeug-ID: INTEGER, Typ: VARCHAR (50), Kapazität: FLOAT, Batteriestatus: INTEGER, Lieferungsnummer: INTEGER(Lieferung)}

Der Fremdschlüssel „Lieferung_Lieferungsnummer“ dient als Verweis auf das Primärschlüsselfeld „Lieferungsnummer“ in der Tabelle „Lieferung“. Bei der Beziehung wird beschrieben, dass jedes Fahrzeug für eine

spezifische Lieferung genutzt wird. Die 1:1-Beziehung hierbei stellt sicher, dass ein Fahrzeug genau einer Lieferung zugeordnet werden kann, und umgekehrt wird jede Lieferung genau durch ein Fahrzeug ausgeführt. Durch diese Art der Verknüpfung wird eine eindeutige Zuordnung sichergestellt, mit der die Nachverfolgbarkeit der Auslieferungslogistik unterstützt wird.

ER-Modell	Relationales Schema								
<p>Lagerhaus</p> <ul style="list-style-type: none"> # * Lager-ID o Standort o Kapazität o Energiequelle 	<p>Lagerhaus</p> <table border="1"> <tr> <td>P * Lager-ID</td> <td>INTEGER</td> </tr> <tr> <td>Standort</td> <td>VARCHAR2 (255)</td> </tr> <tr> <td>Kapazität</td> <td>INTEGER</td> </tr> <tr> <td>Energiequelle</td> <td>VARCHAR2 (4000)</td> </tr> </table> <p>Lagerhaus_PK (Lager-ID)</p>	P * Lager-ID	INTEGER	Standort	VARCHAR2 (255)	Kapazität	INTEGER	Energiequelle	VARCHAR2 (4000)
P * Lager-ID	INTEGER								
Standort	VARCHAR2 (255)								
Kapazität	INTEGER								
Energiequelle	VARCHAR2 (4000)								
Beschreibung									
Lagerhaus: {Lager-ID: INTEGER, Standort: VARCHAR2 (255), Kapazität: INTEGER, Energiequelle: VARCHAR2 (4000)}									

ER-Modell	Relationales Schema								
<p>Artikel</p> <ul style="list-style-type: none"> # * Artikelnummer o Bezeichnung o Gewicht o Lagerbestand 	<p>Artikel</p> <table border="1"> <tr> <td>P * Artikelnummer</td> <td>INTEGER</td> </tr> <tr> <td>Bezeichnung</td> <td>VARCHAR2 (100)</td> </tr> <tr> <td>Gewicht</td> <td>FLOAT</td> </tr> <tr> <td>Lagerbestand</td> <td>INTEGER</td> </tr> </table> <p>F * Lagerhaus_Lager-ID</p> <p>Artikel_PK (Artikelnummer)</p> <p>Artikel_Lagerhaus_FK (Lagerhaus_Lager-ID)</p>	P * Artikelnummer	INTEGER	Bezeichnung	VARCHAR2 (100)	Gewicht	FLOAT	Lagerbestand	INTEGER
P * Artikelnummer	INTEGER								
Bezeichnung	VARCHAR2 (100)								
Gewicht	FLOAT								
Lagerbestand	INTEGER								
Beschreibung									

Artikel: {Artikelnummer: INTEGER, Bezeichnung: VARCHAR2 (100), Gewicht: FLOAT, Lagerbestand: INTEGER, Lager-ID: INTEGER(Lagerhaus)}

Der Fremdschlüssel „Lagerhaus_Lager-ID“ referenziert das Primärschlüssel „Lager-ID“ in der Tabelle „Lagerhaus“. Die 1:n-Beziehung stellt dar, dass ein Lagerhaus mehrere Artikel speichern kann, aber jeder Artikel einem spezifischen Lagerhaus zugeordnet ist. Diese Struktur erlaubt eine klare Zuordnung, welches Lagerhaus für welche Artikel verantwortlich ist, und erleichtert die Verwaltung der Lagerkapazitäten und der Verfügbarkeit von Artikeln.

Relationales Schema – Zwischentabelle „Relation_17“ für die m:n-Beziehung zwischen Lieferung und Artikel				
<p>Relation_17</p> <table border="1"> <tr> <td>PF * Lieferung_Lieferungsnummer</td> <td>INTEGER</td> </tr> <tr> <td>PF * Artikel_Artikelnummer</td> <td>INTEGER</td> </tr> </table> <p>Relation_17_PK (Lieferung_Lieferungsnummer, Artikel_Artikelnummer)</p> <p>Relation_17_Lieferung_FK (Lieferung_Lieferungsnummer)</p> <p>Relation_17_Artikel_FK (Artikel_Artikelnummer)</p>	PF * Lieferung_Lieferungsnummer	INTEGER	PF * Artikel_Artikelnummer	INTEGER
PF * Lieferung_Lieferungsnummer	INTEGER			
PF * Artikel_Artikelnummer	INTEGER			

Der Fremdschlüssel „Lieferung_Lieferungsnummer“ in der Tabelle verweist auf Primärschlüsselfeld „Lieferungsnummer“ in der Tabelle „Lieferung“. Hierdurch wird ermöglicht, dass eine Lieferung mehrere Artikel enthalten kann.

Der Fremdschlüssel „Artikel_Artikelnummer“ in der Tabelle verweist auf das Primärschlüsselfeld „Artikelnummer“ in der Tabelle „Artikel“. Durch diese Verknüpfung kann ein Artikel in mehreren Lieferungen vorkommen. Diese m:n-Beziehung erlaubt die Abbildung komplexer Lieferaufträge, in denen eine Lieferung aus mehreren Artikeln besteht und ein Artikel in mehreren verschiedenen Lieferungen enthalten sein kann.

Stellen Sie die einzelnen Relationen/Tabellen in tabellarischer Form und als CREATE-TABLE-Anweisungen dar. Erzeugen Sie alle Tabellen mit SQL-Befehlen. (Nur CREATE-TABLE-Anweisung verwenden, ohne ALTER TABLE, Index etc...)

Bei dieser Aufgabenstellung sollten wir zu unseren Relationen, diese praktisch umgesetzt, per SQL in Tabellen umsetzen, und hierzu die CREATE-TABLE-Anweisung darstellen. Mit dem Befehl „CREATE TABLE tabellenname“ erstellt man eine Tabelle mit der dazugehörigen Tabellenname.

Kurz vorab: Die Aufgabenstellungen wurden auf dem Server des Informatikstudiengangs der Hochschule Bremerhaven, also auf „hopper“ genauer gesagt auf Mariadb durchgeführt. Mariadb ist ein Open-Source-Datenbankmanagementsystem, ursprünglich als Fork von MySQL entwickelt. Sie bietet eine leistungsfähige, flexible Lösung für Speicherung, Verwaltung und Abfrage von Daten. Mehr dazu hier: <https://www.purestorage.com/de/knowledge/what-is-mariadb.html>

Wurde auch extra im Quellenverzeichnis hinzugefügt.

Erstelle eine Tabelle Lieferung, die Informationen über Lieferungen speichert. Jede Lieferung hat eine Lieferungsnummer als Primärschlüssel, einen Zielort, einen Lieferzeitpunkt und einen Status, der den aktuellen Stand der Lieferung beschreibt.

```
CREATE TABLE Lieferung (
    Lieferungsnummer INT PRIMARY KEY,
    Zielort VARCHAR(500),
    Lieferzeitpunkt DATE,
    Status VARCHAR(255)
);
```

Erstelle eine Tabelle Fahrzeug, die Informationen über verschiedene Fahrzeuge speichert. Jedes Fahrzeug hat eine eindeutige Fahrzeug_ID als Primärschlüssel, einen Typ, der die Art des Fahrzeugs beschreibt, eine Kapazität in Einheiten, einen Batteriestatus als numerischen Wert und eine optionale Lieferungsnummer, die angibt, welcher Lieferung das Fahrzeug zugewiesen wurde. Die Lieferungsnummer ist dabei ein Fremdschlüssel, der auf die Tabelle Lieferung verweist.

```
CREATE TABLE Fahrzeug (
    Fahrzeug_ID INT PRIMARY KEY,
    Typ VARCHAR(50),
    Kapazität FLOAT,
    Batteriestatus INT,
    Lieferung_Lieferungsnummer INT,
    FOREIGN KEY (Lieferung_Lieferungsnummer) REFERENCES Lieferung(Lieferungsnummer)
);
```

Erstelle eine Tabelle Lagerhaus, die Informationen über verschiedene Lagerhäuser speichert. Jedes Lagerhaus hat eine eindeutige Lager_ID als Primärschlüssel, einen Standort, eine maximale Kapazität und eine Energiequelle, die beschreibt, welche Art von Energie zur Versorgung des Lagerhauses genutzt wird.

```
CREATE TABLE Lagerhaus (
    Lager_ID INT PRIMARY KEY,
    Standort VARCHAR(255),
    Kapazität INT,
    Energiequelle VARCHAR(100)
);
```

Erstelle eine Tabelle Artikel, die Informationen über verschiedene Artikel speichert. Jeder Artikel hat eine Artikelnummer als Primärschlüssel, eine Bezeichnung, ein Gewicht, einen Lagerbestand, der die verfügbare Menge angibt, und eine Referenz auf das Lagerhaus, in dem sich der Artikel befindet. Die Lagerhaus_ID ist dabei ein Fremdschlüssel, der auf die Tabelle Lagerhaus verweist.

```
CREATE TABLE Artikel (
    Artikelnummer INT PRIMARY KEY,
    Bezeichnung VARCHAR(100),
    Gewicht FLOAT,
    Lagerbestand INT,
    Lagerhaus_Lager_ID INT,
    FOREIGN KEY (Lagerhaus_Lager_ID) REFERENCES Lagerhaus(Lager_ID)
);
```

Erstelle eine Tabelle Relation_Lieferung_Artikel, die eine n:m-Beziehung zwischen den Tabellen Lieferung und Artikel speichert. Jede Lieferung kann mehrere Artikel enthalten, und jeder Artikel kann in mehreren Lieferungen vorkommen. Die Primärschlüssel sind Lieferungsnummer und Artikelnummer, die gleichzeitig als Fremdschlüssel definiert sind.

```
CREATE TABLE Relation_Lieferung_Artikel (
    Lieferungsnummer INT,
    Artikelnummer INT,
    PRIMARY KEY (Lieferungsnummer, Artikelnummer),
    FOREIGN KEY (Lieferungsnummer) REFERENCES Lieferung(Lieferungsnummer),
    FOREIGN KEY (Artikelnummer) REFERENCES Artikel(Artikelnummer)
);
```

Tables_in_hosakbari_db
Artikel
Fahrzeug
Lagerhaus
Lieferung
Relation_Lieferung_Artikel

Füllen Sie die Datenbank mit einigen Testdaten, verwenden Sie dabei SQL-Befehle (mindestens drei Insert-Operationen pro Tabelle).

Bei dieser Aufgabenstellung sollten in die von uns erstellen Tabellen, Testdaten hineingefügt werden:

Füge eine neue Lieferung mit der Lieferungsnummer 1 in die Tabelle Lieferung ein. Die Lieferung hat als Zielort den Alexanderplatz in Berlin, ist für den 15. Juni 2044 geplant und befindet sich aktuell im Status "unterwegs".

```
INSERT INTO Lieferung (Lieferungsnummer, Zielort, Lieferzeitpunkt, Status)
VALUES (1, 'Berlin, Alexanderplatz', '2044-06-15', 'unterwegs');
```

Füge eine neue Lieferung mit der Lieferungsnummer 2 in die Tabelle Lieferung ein. Die Lieferung hat als Zielort die Hafenstraße in Hamburg, ist für den 16. Juni 2044 vorgesehen und wurde bereits erfolgreich geliefert.

```
INSERT INTO Lieferung (Lieferungsnummer, Zielort, Lieferzeitpunkt, Status)
VALUES (2, 'Hamburg, Hafenstraße', '2044-06-16', 'geliefert');
```

Füge eine neue Lieferung mit der Lieferungsnummer 3 in die Tabelle Lieferung ein. Die Lieferung hat als Zielort den Marienplatz in München, sollte am 17. Juni 2044 zugestellt werden, hat jedoch den Status "verspätet".

```
INSERT INTO Lieferung (Lieferungsnummer, Zielort, Lieferzeitpunkt, Status)
VALUES (3, 'München, Marienplatz', '2044-06-17', 'verspätet');
```

Füge eine neue Lieferung mit der Lieferungsnummer 4 in die Tabelle Lieferung ein. Die Lieferung hat als Zielort die Zeilstraße in Frankfurt, ist für den 18. Juni 2044 geplant und befindet sich aktuell im Status "geplant".

```
INSERT INTO Lieferung (Lieferungsnummer, Zielort, Lieferzeitpunkt, Status)
VALUES (4, 'Frankfurt, Zeilstraße', '2044-06-18', 'geplant');
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status
1	Berlin, Alexanderplatz	2044-06-15	unterwegs
2	Hamburg, Hafenstraße	2044-06-16	geliefert
3	München, Marienplatz	2044-06-17	verspätet
4	Frankfurt, Zeilstraße	2044-06-18	geplant

Ein neues Fahrzeug mit der Fahrzeug_ID 1 wird in die Tabelle Fahrzeug eingefügt. Es handelt sich um einen Elektro-LKW mit einer Kapazität von 5000 Kilogramm. Der Batteriestatus beträgt 85 Prozent. Dieses Fahrzeug ist einer Lieferung mit der Lieferungsnummer 1 zugeordnet.

```
INSERT INTO Fahrzeug (Fahrzeug_ID, Typ, Kapazität, Batteriestatus, Lieferung_Lieferungsnummer)
VALUES (1, 'Elektro-LKW', 5000.0, 85, 1);
```

Ein weiteres Fahrzeug mit der Fahrzeug_ID 2 wird in die Tabelle Fahrzeug eingefügt. Es handelt sich um eine Drohne mit einer Kapazität von 50 Kilogramm. Der Batteriestatus beträgt 100 Prozent. Dieses Fahrzeug ist einer Lieferung mit der Lieferungsnummer 2 zugeordnet.

```
INSERT INTO Fahrzeug (Fahrzeug_ID, Typ, Kapazität, Batteriestatus, Lieferung_Lieferungsnummer)
VALUES (2, 'Drohne', 50.0, 100, 2);
```

Ein weiteres Fahrzeug mit der Fahrzeug_ID 3 wird in die Tabelle Fahrzeug eingefügt. Es handelt sich um einen Elektro-Van mit einer Kapazität von 1000 Kilogramm. Der Batteriestatus beträgt 75 Prozent. Dieses Fahrzeug ist einer Lieferung mit der Lieferungsnummer 3 zugeordnet.

```
INSERT INTO Fahrzeug (Fahrzeug_ID, Typ, Kapazität, Batteriestatus, Lieferung_Lieferungsnummer)
VALUES (3, 'Elektro-Van', 1000.0, 75, 3);
```

Ein weiteres Fahrzeug mit der Fahrzeug_ID 4 wird in die Tabelle Fahrzeug eingefügt. Es handelt sich um einen Hybrid-LKW mit einer Kapazität von 8000 Kilogramm. Der Batteriestatus beträgt 60 Prozent. Dieses Fahrzeug ist derzeit keiner Lieferung zugeordnet, weshalb der Wert in der Spalte Lieferung_Lieferungsnummer auf NULL gesetzt wird.

```
INSERT INTO Fahrzeug (Fahrzeug_ID, Typ, Kapazität, Batteriestatus, Lieferung_Lieferungsnummer)
VALUES (4, 'Hybrid-LKW', 8000.0, 60, NULL);
```

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer
1	Elektro-LKW	5000	85	1
2	Drohne	50	100	2
3	Elektro-Van	1000	75	3
4	Hybrid-LKW	8000	60	NULL

Ein neues Lagerhaus mit der Lager_ID 1 wird in die Tabelle Lagerhaus eingefügt. Es befindet sich in Berlin, in der Lagerstraße 12. Die Kapazität des Lagerhauses beträgt 50000 Einheiten, und als Energiequelle wird Solarenergie genutzt.

```
INSERT INTO Lagerhaus (Lager_ID, Standort, Kapazität, Energiequelle)
VALUES (1, 'Berlin, Lagerstraße 12', 50000, 'Solar');
```

Ein weiteres Lagerhaus mit der Lager_ID 2 wird in die Tabelle Lagerhaus eingefügt. Dieses Lagerhaus befindet sich in Hamburg, im Hafenlager 23. Es verfügt über eine Kapazität von 75000 Einheiten und wird mit Windenergie betrieben.

```
INSERT INTO Lagerhaus (Lager_ID, Standort, Kapazität, Energiequelle)
VALUES (2, 'Hamburg, Hafenlager 23', 75000, 'Wind');
```

Ein weiteres Lagerhaus mit der Lager_ID 3 wird in die Tabelle Lagerhaus eingefügt. Dieses Lagerhaus befindet sich in München, in der Südallee 45. Die Kapazität beträgt 30000 Einheiten, und als Energiequelle wird Solarenergie genutzt.

```
INSERT INTO Lagerhaus (Lager_ID, Standort, Kapazität, Energiequelle)
VALUES (3, 'München, Südallee 45', 30000, 'Solar');
```

Ein weiteres Lagerhaus mit der Lager_ID 4 wird in die Tabelle Lagerhaus eingefügt. Es befindet sich in Frankfurt, im Westhafen 8, und verfügt über eine Kapazität von 100000 Einheiten. Dieses Lagerhaus nutzt eine Kombination aus Solar- und Windenergie als Energiequelle.

```
INSERT INTO Lagerhaus (Lager_ID, Standort, Kapazität, Energiequelle)
VALUES (4, 'Frankfurt, Westhafen 8', 100000, 'Solar/Wind');
```

Lager_ID	Standort	Kapazität	Energiequelle
1	Berlin, Lagerstraße 12	50000	Solar
2	Hamburg, Hafenlager 23	75000	Wind
3	München, Südallee 45	30000	Solar
4	Frankfurt, Westhafen 8	100000	Solar/Wind

Ein neuer Artikel mit der Artikelnummer 101 wird in die Tabelle Artikel eingefügt. Es handelt sich um das Produkt Smartphone-Modell X, das ein Gewicht von 0,2 Kilogramm hat. Der Lagerbestand dieses Artikels beträgt 1500 Einheiten, und er wird im Lagerhaus mit der Lager_ID 1 in Berlin gelagert.

```
INSERT INTO Artikel (Artikelnummer, Bezeichnung, Gewicht, Lagerbestand, Lagerhaus_Lager_ID)
VALUES (101, 'Smartphone-Modell X', 0.2, 1500, 1);
```

Ein weiterer Artikel mit der Artikelnummer 102 wird in die Tabelle Artikel eingefügt. Dieses Produkt ist ein Laptop-Modell Y mit einem Gewicht von 1,5 Kilogramm. Der Lagerbestand beläuft sich auf 500 Einheiten, die im Lagerhaus mit der Lager_ID 2 in Hamburg gelagert werden.

```
INSERT INTO Artikel (Artikelnummer, Bezeichnung, Gewicht, Lagerbestand, Lagerhaus_Lager_ID)
VALUES (102, 'Laptop-Modell Y', 1.5, 500, 2);
```

Ein Artikel mit der Artikelnummer 103 wird ebenfalls in die Tabelle Artikel eingefügt. Es handelt sich um ein Tablet-Modell Z mit einem Gewicht von 0,8 Kilogramm. Der Lagerbestand beträgt 800 Einheiten, die im Lagerhaus mit der Lager_ID 3 in München gelagert werden.

```
INSERT INTO Artikel (Artikelnummer, Bezeichnung, Gewicht, Lagerbestand, Lagerhaus_Lager_ID)
VALUES (103, 'Tablet-Modell Z', 0.8, 800, 3);
```

Ein weiterer Artikel mit der Artikelnummer 104 wird in die Tabelle Artikel eingefügt. Das Produkt Kühlschrank Eco-Kühl hat ein Gewicht von 50 Kilogramm und einen Lagerbestand von 120 Einheiten. Dieses Produkt wird im Lagerhaus mit der Lager_ID 4 in Frankfurt gelagert.

```
INSERT INTO Artikel (Artikelnummer, Bezeichnung, Gewicht, Lagerbestand, Lagerhaus_Lager_ID)
VALUES (104, 'Kühlschrank Eco-Kühl', 50.0, 120, 4);
```

Ein letzter Artikel mit der Artikelnummer 105 wird in die Tabelle Artikel eingefügt. Es handelt sich um eine Waschmaschine SuperClean mit einem Gewicht von 75 Kilogramm. Der Lagerbestand dieses Produkts beträgt 50 Einheiten, die ebenfalls im Lagerhaus mit der Lager_ID 4 in Frankfurt gelagert werden.

```
INSERT INTO Artikel (Artikelnummer, Bezeichnung, Gewicht, Lagerbestand, Lagerhaus_Lager_ID)
VALUES (105, 'Waschmaschine SuperClean', 75.0, 50, 4);
```

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
101	Smartphone-Modell X	0.2	1500	1
102	Laptop-Modell Y	1.5	500	2
103	Tablet-Modell Z	0.8	800	3
104	Kühlschrank Eco-Kühl	50	120	4
105	Waschmaschine SuperClean	75	50	4

In die Tabelle Relation_Lieferung_Artikel wird ein neuer Eintrag eingefügt, der angibt, dass die Lieferung mit der Lieferungsnummer 1 den Artikel mit der Artikelnummer 101 beinhaltet. Ein weiterer Eintrag fügt hinzu, dass dieselbe Lieferung auch den Artikel mit der Artikelnummer 102 enthält.

```
INSERT INTO Relation_Lieferung_Artikel (Lieferungsnummer, Artikelnummer)
VALUES (1, 101);
INSERT INTO Relation_Lieferung_Artikel (Lieferungsnummer, Artikelnummer)
VALUES (1, 102);
```

Ein weiterer Eintrag wird in die Tabelle Relation_Lieferung_Artikel eingefügt, der angibt, dass die Lieferung mit der Lieferungsnummer 2 den Artikel mit der Artikelnummer 103 umfasst. Ebenso wird ein weiterer Eintrag erstellt, der angibt, dass diese Lieferung auch den Artikel mit der Artikelnummer 104 enthält.

```
INSERT INTO Relation_Lieferung_Artikel (Lieferungsnummer, Artikelnummer)
VALUES (2, 103);
INSERT INTO Relation_Lieferung_Artikel (Lieferungsnummer, Artikelnummer)
VALUES (2, 104);
```

Für die Lieferung mit der Lieferungsnummer 3 werden zwei weitere Einträge eingefügt. Der erste gibt an, dass diese Lieferung den Artikel mit der Artikelnummer 105 enthält, und der zweite Eintrag fügt hinzu, dass derselben Lieferung auch der Artikel mit der Artikelnummer 101 zugeordnet ist.

```
INSERT INTO Relation_Lieferung_Artikel (Lieferungsnummer, Artikelnummer)
VALUES (3, 105);
INSERT INTO Relation_Lieferung_Artikel (Lieferungsnummer, Artikelnummer)
VALUES (3, 101);
```

Schließlich werden zwei Einträge für die Lieferung mit der Lieferungsnummer 4 in die Tabelle Relation_Lieferung_Artikel eingefügt. Der erste Eintrag besagt, dass diese Lieferung den Artikel mit der Artikelnummer 102 enthält, während der zweite Eintrag angibt, dass die Lieferung auch den Artikel mit der Artikelnummer 103 umfasst.

```
INSERT INTO Relation_Lieferung_Artikel (Lieferungsnummer, Artikelnummer)
```

```
VALUES (4, 102);
```

```
INSERT INTO Relation_Lieferung_Artikel (Lieferungsnummer, Artikelnummer)
```

```
VALUES (4, 103);
```

Lieferungsnummer	Artikelnummer
1	101
3	101
1	102
4	102
2	103
4	103
2	104
3	105

Führen Sie folgende SQL Operationen: SELECT * FROM TABELLE; SELECT SPALTE1, SPALTE2,... FROM TABELLE; SELECT SPALTE1, SPALTE2,... FROM TABELLE ORDER BY SPALTE1, SPALTE2,...; SELECT SPALTE1, SPALTE2,... FROM TABELLE ORDER BY SPALTE1,... DESC; SELECT * FROM TABELLE WHERE BEDINGUNG; SELECT DISTINCT SPALTE1,... FROM TABELLE

In dieser Aufgabenstellung sollten die oben genannten Befehle für das Auswählen nach Daten aus den Tabellen ausgeführt werden. Diese sind für jede Tabelle anzuwenden:

Artikel – SELECT * FROM Artikel (Lesen aller Spalten einer Tabelle)				
<pre>MariaDB [hosakbari_db]> SELECT * FROM Artikel;</pre>				
Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
101	Smartphone-Modell X	0.2	1500	1
102	Laptop-Modell Y	1.5	500	2
103	Tablet-Modell Z	0.8	800	3
104	Kühlschrank Eco-Kühl	50	120	4
105	Waschmaschine SuperClean	75	50	4

Artikel – SELECT SPALTE{1...5} FROM Artikel (Lesen einzelner Spalten einer Tabelle)				
<pre>MariaDB [hosakbari_db]> SELECT Artikelnummer FROM Artikel;</pre>				
Artikelnummer				
101				
102				
103				
104				
105				

MariaDB [hosakbari_db]> SELECT Bezeichnung FROM Artikel;				
<pre>MariaDB [hosakbari_db]> SELECT Bezeichnung FROM Artikel;</pre>				
Bezeichnung				
Smartphone-Modell X				
Laptop-Modell Y				
Tablet-Modell Z				
Kühlschrank Eco-Kühl				
Waschmaschine SuperClean				

MariaDB [hosakbari_db]> SELECT Gewicht FROM Artikel;				
<pre>MariaDB [hosakbari_db]> SELECT Gewicht FROM Artikel;</pre>				
Gewicht				
0.2				
1.5				
0.8				
50				
75				

MariaDB [hosakbari_db]> SELECT Lagerbestand FROM Artikel;				
<pre>MariaDB [hosakbari_db]> SELECT Lagerbestand FROM Artikel;</pre>				
Lagerbestand				
1500				
500				
800				
120				
50				

```
MariaDB [hosakbari_db]> SELECT Lagerhaus_Lager_ID FROM Artikel;
+-----+
| Lagerhaus_Lager_ID |
+-----+
| 1 |
| 2 |
| 3 |
| 4 |
| 4 |
+-----+
```

Artikel – SELECT SPALTE{1,2; 2,3; 3,4; 4,5} FROM Artikel

```
MariaDB [hosakbari_db]> SELECT Artikelnummer, Bezeichnung FROM Artikel;
+-----+-----+
| Artikelnummer | Bezeichnung |
+-----+-----+
| 101 | Smartphone-Modell X |
| 102 | Laptop-Modell Y |
| 103 | Tablet-Modell Z |
| 104 | Kühlschrank Eco-Kühl |
| 105 | Waschmaschine SuperClean |
+-----+-----+
```

```
MariaDB [hosakbari_db]> SELECT Bezeichnung, Gewicht FROM Artikel;
+-----+-----+
| Bezeichnung | Gewicht |
+-----+-----+
| Smartphone-Modell X | 0.2 |
| Laptop-Modell Y | 1.5 |
| Tablet-Modell Z | 0.8 |
| Kühlschrank Eco-Kühl | 50 |
| Waschmaschine SuperClean | 75 |
+-----+-----+
```

```
MariaDB [hosakbari_db]> SELECT Gewicht, Lagerbestand FROM Artikel;
+-----+-----+
| Gewicht | Lagerbestand |
+-----+-----+
| 0.2 | 1500 |
| 1.5 | 500 |
| 0.8 | 800 |
| 50 | 120 |
| 75 | 50 |
+-----+-----+
```

5 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT Lagerbestand, Lagerhaus_Lager_ID FROM Artikel;
+-----+-----+
| Lagerbestand | Lagerhaus_Lager_ID |
+-----+-----+
| 1500 | 1 |
| 500 | 2 |
| 800 | 3 |
| 120 | 4 |
| 50 | 4 |
+-----+-----+
```

SELECT SPALTE1, SPALTE2, ... FROM Artikel ORDER BY SPALTE1, SPALTE2, ...

Kombination: Einzelne Spalten

Es wird eine Abfrage ausgeführt, um alle Artikelnummern aus der Tabelle Artikel abzufragen und diese aufsteigend nach ihrer Artikelnummer zu sortieren.

```
SELECT Artikelnummer FROM Artikel ORDER BY Artikelnummer;
```

Anschließend wird eine weitere Abfrage durchgeführt, die alle Bezeichnungen der Artikel aus der Tabelle Artikel abruft und diese alphabetisch sortiert.

```
SELECT Bezeichnung FROM Artikel ORDER BY Bezeichnung;
```

Eine zusätzliche Abfrage liefert das Gewicht aller Artikel aus der Tabelle Artikel und sortiert die Ergebnisse aufsteigend nach dem Gewicht.

```
SELECT Gewicht FROM Artikel ORDER BY Gewicht;
```

Zuletzt wird eine Abfrage ausgeführt, die den Lagerbestand aller Artikel aus der Tabelle Artikel ermittelt und diese Werte aufsteigend nach der Anzahl im Lagerbestand sortiert. Abschließend wird eine Abfrage erstellt, die alle Lagerhaus-IDs der Artikel anzeigt und diese aufsteigend nach der Lagerhaus-ID sortiert.

```
SELECT Lagerbestand FROM Artikel ORDER BY Lagerbestand;
```

```
SELECT Lagerhaus_Lager_ID FROM Artikel ORDER BY Lagerhaus_Lager_ID;
```

```
MariaDB [hosakbari_db]> SELECT Artikelnummer FROM Artikel ORDER BY Artikelnummer;
```

Artikelnummer
101
102
103
104
105

```
MariaDB [hosakbari_db]> SELECT Bezeichnung FROM Artikel ORDER BY Bezeichnung;
```

Bezeichnung
Kühlschrank Eco-Kühl
Laptop-Modell Y
Smartphone-Modell X
Tablet-Modell Z
Waschmaschine SuperClean

```
MariaDB [hosakbari_db]> SELECT Gewicht FROM Artikel ORDER BY Gewicht;
```

Gewicht
0.2
0.8
1.5
50
75

```
MariaDB [hosakbari_db]> SELECT Lagerbestand FROM Artikel ORDER BY Lagerbestand;
```

Lagerbestand
50
120
500
800
1500

Lagerhaus_Lager_ID
1
2
3
4
4

Kombination: Zwei Spalten

Es wird eine Abfrage erstellt, die die Artikelnummer und die Bezeichnung aller Artikel aus der Tabelle Artikel abruft und die Ergebnisse zuerst nach der Artikelnummer und dann nach der Bezeichnung sortiert.

SELECT Artikelnummer, Bezeichnung FROM Artikel ORDER BY Artikelnummer, Bezeichnung;

Eine weitere Abfrage liefert die Artikelnummer und das Gewicht der Artikel, wobei die Ergebnisse zuerst nach der Artikelnummer und anschließend nach dem Gewicht sortiert werden.

SELECT Artikelnummer, Gewicht FROM Artikel ORDER BY Artikelnummer, Gewicht;

Eine zusätzliche Abfrage zeigt die Artikelnummer und den Lagerbestand der Artikel an, sortiert nach Artikelnummer und Lagerbestand.

SELECT Artikelnummer, Lagerbestand FROM Artikel ORDER BY Artikelnummer, Lagerbestand;

Es folgt eine Abfrage, die die Artikelnummer und die Lagerhaus-ID jedes Artikels abruft, sortiert nach Artikelnummer und Lagerhaus-ID.

SELECT Artikelnummer, Lagerhaus_Lager_ID FROM Artikel ORDER BY Artikelnummer, Lagerhaus_Lager_ID;

Anschließend wird eine Abfrage durchgeführt, die die Bezeichnung und das Gewicht der Artikel liefert, wobei die Ergebnisse alphabetisch nach der Bezeichnung und anschließend nach dem Gewicht sortiert werden.

SELECT Bezeichnung, Gewicht FROM Artikel ORDER BY Bezeichnung, Gewicht;

Eine weitere Abfrage zeigt die Bezeichnung und den Lagerbestand der Artikel an, sortiert nach der Bezeichnung und dem Lagerbestand.

SELECT Bezeichnung, Lagerbestand FROM Artikel ORDER BY Bezeichnung, Lagerbestand;

Eine Abfrage listet die Bezeichnung und die Lagerhaus-ID der Artikel auf, geordnet nach Bezeichnung und Lagerhaus-ID.

SELECT Bezeichnung, Lagerhaus_Lager_ID FROM Artikel ORDER BY Bezeichnung, Lagerhaus_Lager_ID;

Zusätzlich wird eine Abfrage erstellt, die das Gewicht und den Lagerbestand der Artikel abruft und die Ergebnisse nach Gewicht und Lagerbestand sortiert.

SELECT Gewicht, Lagerbestand FROM Artikel ORDER BY Gewicht, Lagerbestand;

Eine weitere Abfrage zeigt das Gewicht und die Lagerhaus-ID der Artikel, sortiert nach Gewicht und Lagerhaus-ID.

SELECT Gewicht, Lagerhaus_Lager_ID FROM Artikel ORDER BY Gewicht, Lagerhaus_Lager_ID;

Abschließend wird eine Abfrage ausgeführt, die den Lagerbestand und die Lagerhaus-ID der Artikel abruft und die Ergebnisse nach Lagerbestand und Lagerhaus-ID sortiert.

```

SELECT Lagerbestand, Lagerhaus_Lager_ID FROM Artikel ORDER BY Lagerbestand, Lagerhaus_Lager_ID;
MariaDB [hosakbari_db]> SELECT Artikelnummer, Bezeichnung FROM Artikel ORDER BY Artikelnummer, Bezeichnung;
+-----+-----+
| Artikelnummer | Bezeichnung |
+-----+-----+
| 101 | Smartphone-Modell X |
| 102 | Laptop-Modell Y |
| 103 | Tablet-Modell Z |
| 104 | Kühlschrank Eco-Kühl |
| 105 | Waschmaschine SuperClean |
+-----+-----+
MariaDB [hosakbari_db]> SELECT Artikelnummer, Gewicht FROM Artikel ORDER BY Artikelnummer, Gewicht;
+-----+-----+
| Artikelnummer | Gewicht |
+-----+-----+
| 101 | 0.2 |
| 102 | 1.5 |
| 103 | 0.8 |
| 104 | 50 |
| 105 | 75 |
+-----+-----+
MariaDB [hosakbari_db]> SELECT Artikelnummer, Lagerbestand FROM Artikel ORDER BY Artikelnummer, Lagerbestand;
+-----+-----+
| Artikelnummer | Lagerbestand |
+-----+-----+
| 101 | 1500 |
| 102 | 500 |
| 103 | 800 |
| 104 | 120 |
| 105 | 50 |
+-----+-----+
MariaDB [hosakbari_db]> SELECT Artikelnummer, Lagerhaus_Lager_ID FROM Artikel ORDER BY Artikelnummer, Lagerhaus_Lager_ID;
+-----+-----+
| Artikelnummer | Lagerhaus_Lager_ID |
+-----+-----+
| 101 | 1 |
| 102 | 2 |
| 103 | 3 |
| 104 | 4 |
| 105 | 4 |
+-----+-----+
MariaDB [hosakbari_db]> SELECT Bezeichnung, Gewicht FROM Artikel ORDER BY Bezeichnung, Gewicht;
+-----+-----+
| Bezeichnung | Gewicht |
+-----+-----+
| Kühlschrank Eco-Kühl | 50 |
| Laptop-Modell Y | 1.5 |
| Smartphone-Modell X | 0.2 |
| Tablet-Modell Z | 0.8 |
| Waschmaschine SuperClean | 75 |
+-----+-----+
MariaDB [hosakbari_db]> SELECT Bezeichnung, Lagerbestand FROM Artikel ORDER BY Bezeichnung, Lagerbestand;
+-----+-----+
| Bezeichnung | Lagerbestand |
+-----+-----+
| Kühlschrank Eco-Kühl | 120 |
| Laptop-Modell Y | 500 |
| Smartphone-Modell X | 1500 |
| Tablet-Modell Z | 800 |
| Waschmaschine SuperClean | 50 |
+-----+-----+
MariaDB [hosakbari_db]> SELECT Bezeichnung, Lagerhaus_Lager_ID FROM Artikel ORDER BY Bezeichnung, Lagerhaus_Lager_ID;
+-----+-----+
| Bezeichnung | Lagerhaus_Lager_ID |
+-----+-----+
| Kühlschrank Eco-Kühl | 4 |
| Laptop-Modell Y | 2 |
| Smartphone-Modell X | 1 |
| Tablet-Modell Z | 3 |
| Waschmaschine SuperClean | 4 |
+-----+-----+

```

```
MariaDB [hosakbari_db]> SELECT Gewicht, Lagerbestand FROM Artikel ORDER BY Gewicht, Lagerbestand;
+-----+-----+
| Gewicht | Lagerbestand |
+-----+-----+
| 0.2     | 1500   |
| 0.8     | 800    |
| 1.5     | 500    |
| 50      | 120    |
| 75      | 50     |
+-----+-----+
MariaDB [hosakbari_db]> SELECT Gewicht, Lagerhaus_Lager_ID FROM Artikel ORDER BY Gewicht, Lagerhaus_Lager_ID;
+-----+-----+
| Gewicht | Lagerhaus_Lager_ID |
+-----+-----+
| 0.2     | 1       |
| 0.8     | 3       |
| 1.5     | 2       |
| 50      | 4       |
| 75      | 4       |
+-----+-----+
MariaDB [hosakbari_db]> SELECT Lagerbestand, Lagerhaus_Lager_ID FROM Artikel ORDER BY Lagerbestand, Lagerhaus_Lager_ID;
+-----+-----+
| Lagerbestand | Lagerhaus_Lager_ID |
+-----+-----+
| 50          | 4       |
| 120         | 4       |
| 500         | 2       |
| 800         | 3       |
| 1500        | 1       |
+-----+-----+
```

SELECT SPALTE1, SPALTE2, ... FROM Artikel ORDER BY SPALTE1, ... DESC (Absteigendes Sortieren des Spalteninhalts)

Kombination: Einzelne Spalte:

Es wird eine Abfrage erstellt, die alle Artikelnummern aus der Tabelle Artikel abruft und die Ergebnisse in absteigender Reihenfolge sortiert.

SELECT Artikelnummer FROM Artikel ORDER BY Artikelnummer DESC;

Eine weitere Abfrage liefert die Bezeichnungen aller Artikel, wobei die Ergebnisse in absteigender alphabetischer Reihenfolge sortiert werden.

SELECT Bezeichnung FROM Artikel ORDER BY Bezeichnung DESC;

Es folgt eine Abfrage, die die Gewichte aller Artikel abruft und diese in absteigender Reihenfolge sortiert.

SELECT Gewicht FROM Artikel ORDER BY Gewicht DESC;

Eine zusätzliche Abfrage zeigt den Lagerbestand aller Artikel an, sortiert in absteigender Reihenfolge.

SELECT Lagerbestand FROM Artikel ORDER BY Lagerbestand DESC;

Abschließend wird eine Abfrage ausgeführt, die die Lagerhaus-IDs aller Artikel liefert und die Ergebnisse in absteigender Reihenfolge sortiert.

SELECT Lagerhaus_Lager_ID FROM Artikel ORDER BY Lagerhaus_Lager_ID DESC;

```
MariaDB [hosakbari_db]> SELECT Artikelnummer FROM Artikel ORDER BY Artikelnummer DESC
+-----+
| Artikelnummer |
+-----+
|      105   |
|      104   |
|      103   |
|      102   |
|      101   |
+-----+
MariaDB [hosakbari_db]> SELECT Bezeichnung FROM Artikel ORDER BY Bezeichnung DESC
+-----+
| Bezeichnung |
+-----+
| Waschmaschine SuperClean |
| Tablet-Modell Z          |
| Smartphone-Modell X     |
| Laptop-Modell Y          |
| Kühlschrank Eco-Kühl     |
+-----+
MariaDB [hosakbari_db]> SELECT Gewicht FROM Artikel ORDER BY Gewicht DESC;
+-----+
| Gewicht |
+-----+
|      75   |
|      50   |
|      1.5  |
|      0.8  |
|      0.2  |
+-----+
MariaDB [hosakbari_db]> SELECT Lagerbestand FROM Artikel ORDER BY Lagerbestand DESC;
+-----+
| Lagerbestand |
+-----+
|     1500   |
|     800    |
|     500    |
|     120    |
|      50   |
+-----+
MariaDB [hosakbari_db]> SELECT Lagerhaus_Lager_ID FROM Artikel ORDER BY Lagerhaus_Lager_ID DESC;
+-----+
| Lagerhaus_Lager_ID |
+-----+
|       4   |
|       4   |
|       3   |
|       2   |
|       1   |
+-----+
```

Kombination: Zwei Spalten:

Es wird eine Abfrage erstellt, die die Artikelnummern und Bezeichnungen aller Artikel abruft, wobei die Ergebnisse sowohl nach Artikelnummer als auch nach Bezeichnung in absteigender Reihenfolge sortiert werden.

```
SELECT Artikelnummer, Bezeichnung FROM Artikel ORDER BY Artikelnummer DESC, Bezeichnung DESC;
```

Eine weitere Abfrage gibt die Artikelnummern und Gewichte der Artikel aus, sortiert nach beiden Attributen in absteigender Reihenfolge.

```
SELECT Artikelnummer, Gewicht FROM Artikel ORDER BY Artikelnummer DESC, Gewicht DESC;
```

Die nächste Abfrage zeigt die Artikelnummern und den Lagerbestand aller Artikel, ebenfalls sortiert in absteigender Reihenfolge nach beiden Werten.

```
SELECT Artikelnummer, Lagerbestand FROM Artikel ORDER BY Artikelnummer DESC, Lagerbestand DESC;
```

Es folgt eine Abfrage, die die Artikelnummern und die Lagerhaus-IDs der Artikel liefert, sortiert in absteigender Reihenfolge.

```
SELECT Artikelnummer, Lagerhaus_Lager_ID FROM Artikel ORDER BY Artikelnummer DESC, Lagerhaus_Lager_ID DESC;
```

Zusätzlich wird eine Abfrage erstellt, die die Bezeichnungen und Gewichte der Artikel abruft, sortiert in absteigender Reihenfolge nach beiden Spalten.

```
SELECT Bezeichnung, Gewicht FROM Artikel ORDER BY Bezeichnung DESC, Gewicht DESC;
```

Eine weitere Abfrage gibt die Bezeichnungen und den Lagerbestand der Artikel zurück, sortiert in absteigender Reihenfolge.

```
SELECT Bezeichnung, Lagerbestand FROM Artikel ORDER BY Bezeichnung DESC, Lagerbestand DESC;
```

Es wird eine Abfrage erstellt, die die Bezeichnungen und die Lagerhaus-IDs der Artikel liefert, ebenfalls sortiert in absteigender Reihenfolge.

```
SELECT Bezeichnung, Lagerhaus_Lager_ID FROM Artikel ORDER BY Bezeichnung DESC, Lagerhaus_Lager_ID DESC;
```

Eine weitere Abfrage zeigt die Gewichte und den Lagerbestand der Artikel, sortiert in absteigender Reihenfolge nach beiden Werten.

```
SELECT Gewicht, Lagerbestand FROM Artikel ORDER BY Gewicht DESC, Lagerbestand DESC;
```

Es folgt eine Abfrage, die die Gewichte und die Lagerhaus-IDs der Artikel abruft, sortiert in absteigender Reihenfolge.

```
SELECT Gewicht, Lagerhaus_Lager_ID FROM Artikel ORDER BY Gewicht DESC, Lagerhaus_Lager_ID DESC;
```

Abschließend wird eine Abfrage ausgeführt, die den Lagerbestand und die Lagerhaus-IDs der Artikel liefert, sortiert in absteigender Reihenfolge.

```
SELECT Lagerbestand, Lagerhaus_Lager_ID FROM Artikel ORDER BY Lagerbestand DESC, Lagerhaus_Lager_ID DESC;
```

MariaDB [hosakbari_db]> SELECT Artikelnummer, Bezeichnung FROM Artikel ORDER BY Artikelnummer DESC, Bezeichnung DESC;	
Artikelnummer	Bezeichnung
105	Waschmaschine SuperClean
104	Kühlschrank Eco-Kühl
103	Tablet-Modell Z
102	Laptop-Modell Y
101	Smartphone-Modell X

MariaDB [hosakbari_db]> SELECT Artikelnummer, Gewicht FROM Artikel ORDER BY Artikelnummer DESC, Gewicht DESC;	
Artikelnummer	Gewicht
105	75
104	50
103	0.8
102	1.5
101	0.2

```
MariaDB [hosakbari_db]> SELECT Artikelnummer, Lagerbestand FROM Artikel ORDER BY Artikelnummer DESC, Lagerbestand DESC;
+-----+-----+
| Artikelnummer | Lagerbestand |
+-----+-----+
| 105 | 50 |
| 104 | 120 |
| 103 | 800 |
| 102 | 500 |
| 101 | 1500 |
+-----+-----+
MariaDB [hosakbari_db]> SELECT Artikelnummer, Lagerhaus_Lager_ID FROM Artikel ORDER BY Artikelnummer DESC, Lagerhaus_Lager_ID DESC;
+-----+-----+
| Artikelnummer | Lagerhaus_Lager_ID |
+-----+-----+
| 105 | 4 |
| 104 | 4 |
| 103 | 3 |
| 102 | 2 |
| 101 | 1 |
+-----+-----+
MariaDB [hosakbari_db]> SELECT Bezeichnung, Gewicht FROM Artikel ORDER BY Bezeichnung DESC, Gewicht DESC;
+-----+-----+
| Bezeichnung | Gewicht |
+-----+-----+
| Waschmaschine SuperClean | 75 |
| Tablet-Modell Z | 0.8 |
| Smartphone-Modell X | 0.2 |
| Laptop-Modell Y | 1.5 |
| Kühlschrank Eco-Kühl | 50 |
+-----+-----+
MariaDB [hosakbari_db]> SELECT Bezeichnung, Lagerbestand FROM Artikel ORDER BY Bezeichnung DESC, Lagerbestand DESC;
+-----+-----+
| Bezeichnung | Lagerbestand |
+-----+-----+
| Waschmaschine SuperClean | 50 |
| Tablet-Modell Z | 800 |
| Smartphone-Modell X | 1500 |
| Laptop-Modell Y | 500 |
| Kühlschrank Eco-Kühl | 120 |
+-----+-----+
MariaDB [hosakbari_db]> SELECT Bezeichnung, Lagerhaus_Lager_ID FROM Artikel ORDER BY Bezeichnung DESC, Lagerhaus_Lager_ID DESC;
+-----+-----+
| Bezeichnung | Lagerhaus_Lager_ID |
+-----+-----+
| Waschmaschine SuperClean | 4 |
| Tablet-Modell Z | 3 |
| Smartphone-Modell X | 1 |
| Laptop-Modell Y | 2 |
| Kühlschrank Eco-Kühl | 4 |
+-----+-----+
MariaDB [hosakbari_db]> SELECT Gewicht, Lagerbestand FROM Artikel ORDER BY Gewicht DESC, Lagerbestand DESC;
+-----+-----+
| Gewicht | Lagerbestand |
+-----+-----+
| 75 | 50 |
| 50 | 120 |
| 1.5 | 500 |
| 0.8 | 800 |
| 0.2 | 1500 |
+-----+-----+
MariaDB [hosakbari_db]> SELECT Gewicht, Lagerhaus_Lager_ID FROM Artikel ORDER BY Gewicht DESC, Lagerhaus_Lager_ID DESC;
+-----+-----+
| Gewicht | Lagerhaus_Lager_ID |
+-----+-----+
| 75 | 4 |
| 50 | 4 |
| 1.5 | 2 |
| 0.8 | 3 |
| 0.2 | 1 |
+-----+-----+
```

Lagerbestand	Lagerhaus_Lager_ID
1500	1
800	3
500	2
120	4
50	4

SELECT * FROM TABELLE WHERE BEDINGUNG (Selektives Lesen einzelner Zeilen)

Beispiele mit einer einfachen Bedingung:

Es wird eine Abfrage durchgeführt, um alle Datensätze aus der Tabelle „Artikel“ abzurufen, bei denen die Artikelnummer genau 101 ist.

SELECT * FROM Artikel WHERE Artikelnummer = 101;

Eine weitere Abfrage gibt alle Artikel aus, die im Lagerhaus mit der Lager-ID 4 gespeichert sind.

SELECT * FROM Artikel WHERE Lagerhaus_Lager_ID = 4;

Es folgt eine Abfrage, die alle Artikel liefert, deren Lagerbestand größer als 1000 Einheiten ist.

SELECT * FROM Artikel WHERE Lagerbestand > 1000;

Eine weitere Abfrage listet alle Artikel auf, deren Gewicht weniger als 1 Kilogramm beträgt.

SELECT * FROM Artikel WHERE Gewicht < 1;

Abschließend wird eine Abfrage erstellt, um alle Artikel anzuzeigen, deren Bezeichnung das Wort „Modell“ enthält.

SELECT * FROM Artikel WHERE Bezeichnung LIKE '%Modell%';

Suche nach einer bestimmten Artikelnummer:

SELECT * FROM Artikel WHERE Artikelnummer = 101;

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
101	Smartphone-Modell X	0.2	1500	1

Suche nach Artikeln in einem bestimmten Lagerhaus:

SELECT * FROM Artikel WHERE Lagerhaus_Lager_ID = 4;

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
104	Kühlschrank Eco-Kühl	50	120	4
105	Waschmaschine SuperClean	75	50	4

Suche nach Artikeln mit einem Lagerbestand größer als 1000:

SELECT * FROM Artikel WHERE Lagerbestand > 1000;

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
101	Smartphone-Modell X	0.2	1500	1

Suche nach Artikeln mit einem Gewicht kleiner als 1 kg:

SELECT * FROM Artikel WHERE Gewicht < 1;

MariaDB [hosakbari_db]> SELECT * FROM Artikel WHERE Gewicht < 1;				
Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
101	Smartphone-Modell X	0.2	1500	1
103	Tablet-Modell Z	0.8	800	3

Suche nach Artikeln mit einem bestimmten Bezeichnungswort:

SELECT * FROM Artikel WHERE Bezeichnung LIKE '%Modell%';

MariaDB [hosakbari_db]> SELECT * FROM Artikel WHERE Bezeichnung LIKE '%Modell%';				
Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
101	Smartphone-Modell X	0.2	1500	1
102	Laptop-Modell Y	1.5	500	2
103	Tablet-Modell Z	0.8	800	3

Beispiele mit mehreren Bedingungen:

Es wird eine Abfrage erstellt, die alle Artikel auflistet, deren Gewicht weniger als 1 Kilogramm beträgt und deren Lagerbestand mehr als 500 Einheiten umfasst.

SELECT * FROM Artikel WHERE Gewicht < 1 AND Lagerbestand > 500;

Eine weitere Abfrage zeigt alle Artikel, die im Lagerhaus mit der Lager-ID 4 gespeichert sind und deren Lagerbestand weniger als 100 Einheiten beträgt.

SELECT * FROM Artikel WHERE Lagerhaus_Lager_ID = 4 AND Lagerbestand < 100;

Es folgt eine Abfrage, die alle Artikel auflistet, deren Bezeichnung mit „Eco“ endet und deren Gewicht mehr als 50 Kilogramm beträgt.

SELECT * FROM Artikel WHERE Bezeichnung LIKE '%Eco' AND Gewicht > 50;

Abschließend wird eine Abfrage durchgeführt, um alle Artikel anzuzeigen, deren Lagerbestand zwischen 100 und 1000 Einheiten liegt.

SELECT * FROM Artikel WHERE Lagerbestand BETWEEN 100 AND 1000;

Suche nach Artikeln mit einem Gewicht kleiner als 1 kg und einem Lagerbestand größer als 500:

SELECT * FROM Artikel WHERE Gewicht < 1 AND Lagerbestand > 500;

MariaDB [hosakbari_db]> SELECT * FROM Artikel WHERE Gewicht < 1 AND Lagerbestand > 500;				
Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
101	Smartphone-Modell X	0.2	1500	1
103	Tablet-Modell Z	0.8	800	3

Suche nach Artikeln in einem bestimmten Lagerhaus mit einem Lagerbestand kleiner als 100:

SELECT * FROM Artikel WHERE Lagerhaus_Lager_ID = 4 AND Lagerbestand < 100;

MariaDB [hosakbari_db]> SELECT * FROM Artikel WHERE Lagerhaus_Lager_ID = 4 AND Lagerbestand < 100;				
Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
105	Waschmaschine SuperClean	75	50	4

Suche nach Artikeln mit einem Lagerbestand zwischen 100 und 1000:

SELECT * FROM Artikel WHERE Lagerbestand BETWEEN 100 AND 1000;

MariaDB [hosakbari_db]> SELECT * FROM Artikel WHERE Lagerbestand BETWEEN 100 AND 1000;				
Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
102	Laptop-Modell Y	1.5	500	2
103	Tablet-Modell Z	0.8	800	3
104	Kühlschrank Eco-Kühl	50	120	4

Beispiele mit OR:

Es wird eine Abfrage erstellt, die alle Artikel anzeigt, deren Artikelnummer entweder 101 oder 102 ist.

```
SELECT * FROM Artikel WHERE Artikelnummer = 101 OR Artikelnummer = 102;
```

Eine weitere Abfrage gibt alle Artikel zurück, die sich entweder im Lagerhaus mit der Lager-ID 1 oder im Lagerhaus mit der Lager-ID 3 befinden.

```
SELECT * FROM Artikel WHERE Lagerhaus_Lager_ID = 1 OR Lagerhaus_Lager_ID = 3;
```

Zum Schluss wird eine Abfrage durchgeführt, um alle Artikel aufzulisten, deren Gewicht mehr als 70 Kilogramm beträgt oder deren Lagerbestand weniger als 200 Einheiten umfasst.

```
SELECT * FROM Artikel WHERE Gewicht > 70 OR Lagerbestand < 200;
```

Suche nach Artikeln mit einer Artikelnummer von 101 oder 102:

```
SELECT * FROM Artikel WHERE Artikelnummer = 101 OR Artikelnummer = 102;
```

MariaDB [hosakbari_db]> SELECT * FROM Artikel WHERE Artikelnummer = 101 OR Artikelnummer = 102				
Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
101	Smartphone-Modell X	0.2	1500	1
102	Laptop-Modell Y	1.5	500	2

Suche nach Artikeln, die entweder in Lagerhaus 1 oder in Lagerhaus 3 gelagert sind:

```
SELECT * FROM Artikel WHERE Lagerhaus_Lager_ID = 1 OR Lagerhaus_Lager_ID = 3;
```

MariaDB [hosakbari_db]> SELECT * FROM Artikel WHERE Lagerhaus_Lager_ID = 1 OR Lagerhaus_Lager_ID = 3				
Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
101	Smartphone-Modell X	0.2	1500	1
103	Tablet-Modell Z	0.8	800	3

Suche nach Artikeln mit einem Gewicht größer als 70 kg oder einem Lagerbestand kleiner als 200:

```
SELECT * FROM Artikel WHERE Gewicht > 70 OR Lagerbestand < 200;
```

MariaDB [hosakbari_db]> SELECT * FROM Artikel WHERE Gewicht > 70 OR Lagerbestand < 200;				
Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
104	Kühlschrank Eco-Kühl	50	120	4
105	Waschmaschine SuperClean	75	50	4

Beispiele mit NOT

Es wird eine Abfrage erstellt, die alle Artikel ausgibt, die sich nicht im Lagerhaus mit der Lager-ID 2 befinden.

```
SELECT * FROM Artikel WHERE NOT Lagerhaus_Lager_ID = 2;
```

Eine weitere Abfrage zeigt alle Artikel, deren Bezeichnung nicht „Smartphone-Modell X“ ist.

```
SELECT * FROM Artikel WHERE NOT Bezeichnung = 'Smartphone-Modell X';
```

Zum Schluss wird eine Abfrage durchgeführt, die alle Artikel auflistet, deren Gewicht nicht zwischen 0,5 und 1,5 Kilogramm liegt.

```
SELECT * FROM Artikel WHERE NOT (Gewicht BETWEEN 0.5 AND 1.5);
```

Suche nach Artikeln, die nicht in Lagerhaus 2 gelagert sind:

```
SELECT * FROM Artikel WHERE NOT Lagerhaus_Lager_ID = 2;
```

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
101	Smartphone-Modell X	0.2	1500	1
103	Tablet-Modell Z	0.8	800	3
104	Kühlschrank Eco-Kühl	50	120	4
105	Waschmaschine SuperClean	75	50	4

Suche nach Artikeln, deren Bezeichnung nicht "Smartphone-Modell X" ist:

```
SELECT * FROM Artikel WHERE NOT Bezeichnung = 'Smartphone-Modell X';
```

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
102	Laptop-Modell Y	1.5	500	2
103	Tablet-Modell Z	0.8	800	3
104	Kühlschrank Eco-Kühl	50	120	4
105	Waschmaschine SuperClean	75	50	4

Suche nach Artikeln mit einem Gewicht nicht zwischen 0.5 und 1.5 kg:

```
SELECT * FROM Artikel WHERE NOT (Gewicht BETWEEN 0.5 AND 1.5);
```

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
101	Smartphone-Modell X	0.2	1500	1
104	Kühlschrank Eco-Kühl	50	120	4
105	Waschmaschine SuperClean	75	50	4

Beispiele mit IN:

Es wird eine Abfrage erstellt, die alle Artikel anzeigt, die sich in den Lagerhäusern mit den Lager-IDs 1, 2 oder 4 befinden.

```
SELECT * FROM Artikel WHERE Lagerhaus_Lager_ID IN (1, 2, 4);
```

Eine weitere Abfrage zeigt alle Artikel, deren Artikelnummer 101, 103 oder 105 ist.

```
SELECT * FROM Artikel WHERE Artikelnummer IN (101, 103, 105);
```

Suche nach Artikeln, die in Lagerhaus 1, 2 oder 4 gelagert sind:

```
SELECT * FROM Artikel WHERE Lagerhaus_Lager_ID IN (1, 2, 4);
```

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
101	Smartphone-Modell X	0.2	1500	1
102	Laptop-Modell Y	1.5	500	2
104	Kühlschrank Eco-Kühl	50	120	4
105	Waschmaschine SuperClean	75	50	4

Suche nach Artikeln mit Artikelnummern 101, 103 oder 105:

```
SELECT * FROM Artikel WHERE Artikelnummer IN (101, 103, 105);
```

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
101	Smartphone-Modell X	0.2	1500	1
103	Tablet-Modell Z	0.8	800	3
105	Waschmaschine SuperClean	75	50	4

Beispiele mit ORDER BY und WHERE:

Es wird eine Abfrage erstellt, die alle Artikel anzeigt, deren Lagerbestand größer als 500 ist, sortiert nach ihrem Gewicht in aufsteigender Reihenfolge.

```
SELECT * FROM Artikel WHERE Lagerbestand > 500 ORDER BY Gewicht ASC;
```

Eine weitere Abfrage listet alle Artikel auf, deren Gewicht unter 1 liegt, und sortiert sie in absteigender Reihenfolge nach ihrer Bezeichnung.

```
SELECT * FROM Artikel WHERE Gewicht < 1 ORDER BY Bezeichnung DESC;
```

Suche nach Artikeln mit einem Lagerbestand größer als 500, sortiert nach Gewicht:

```
SELECT * FROM Artikel WHERE Lagerbestand > 500 ORDER BY Gewicht ASC;
```

MariaDB [hosakbari_db]> SELECT * FROM Artikel WHERE Lagerbestand > 500 ORDER BY Gewicht ASC;				
Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
101	Smartphone-Modell X	0.2	1500	1
103	Tablet-Modell Z	0.8	800	3

Suche nach Artikeln mit einem Gewicht kleiner als 1 kg, sortiert nach Bezeichnung:

```
SELECT * FROM Artikel WHERE Gewicht < 1 ORDER BY Bezeichnung DESC;
```

MariaDB [hosakbari_db]> SELECT * FROM Artikel WHERE Gewicht < 1 ORDER BY Bezeichnung DESC;				
Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
103	Tablet-Modell Z	0.8	800	3
101	Smartphone-Modell X	0.2	1500	1

SELECT DISTINCT SPALTE1, ... FROM Artikel (In der Ergebnistabelle keine doppelten Zeilen)

Beispiele mit einer Spalte:

Es wird eine Abfrage erstellt, die alle eindeutigen Lagerhaus-IDs anzeigt, in denen Artikel gelagert sind.

```
SELECT DISTINCT Lagerhaus_Lager_ID FROM Artikel;
```

Eine weitere Abfrage zeigt alle eindeutigen Bezeichnungen der Artikel in der Datenbank.

```
SELECT DISTINCT Bezeichnung FROM Artikel;
```

Schließlich listet die letzte Abfrage alle unterschiedlichen Gewichtsangaben der Artikel auf.

```
SELECT DISTINCT Gewicht FROM Artikel;
```

Zeige alle unterschiedlichen Lagerhaus-IDs, in denen Artikel gelagert sind:

```
SELECT DISTINCT Lagerhaus_Lager_ID FROM Artikel;
```

MariaDB [hosakbari_db]> SELECT DISTINCT Lagerhaus_Lager_ID FROM Artikel;	
Lagerhaus_Lager_ID	
1	
2	
3	
4	

Zeige alle unterschiedlichen Artikelbezeichnungen:

```
SELECT DISTINCT Bezeichnung FROM Artikel;
```

```
MariaDB [hosakbari_db]> SELECT DISTINCT Bezeichnung FROM Artikel;
+-----+
| Bezeichnung |
+-----+
| Smartphone-Modell X |
| Laptop-Modell Y |
| Tablet-Modell Z |
| Kühlschrank Eco-Kühl |
| Waschmaschine SuperClean |
+-----+
```

Zeige alle unterschiedlichen Gewichtswerte:

```
SELECT DISTINCT Gewicht FROM Artikel;
```

```
MariaDB [hosakbari_db]> SELECT DISTINCT Gewicht FROM Artikel;
+-----+
| Gewicht |
+-----+
| 0.2 |
| 1.5 |
| 0.8 |
| 50 |
| 75 |
+-----+
```

Beispiele mit mehreren Spalten:

Es wird eine Abfrage erstellt, die alle eindeutigen Kombinationen aus Lagerhaus-IDs und Artikelbezeichnungen anzeigt.

```
SELECT DISTINCT Lagerhaus_Lager_ID, Bezeichnung FROM Artikel;
```

Eine weitere Abfrage zeigt alle eindeutigen Kombinationen aus Gewicht und Lagerbestand der Artikel.

```
SELECT DISTINCT Gewicht, Lagerbestand FROM Artikel;
```

Die letzte Abfrage listet alle unterschiedlichen Kombinationen aus Artikelnummern und Bezeichnungen der Artikel auf.

```
SELECT DISTINCT Artikelnummer, Bezeichnung FROM Artikel;
```

Zeige alle Kombinationen aus Lagerhaus-IDs und Artikelbezeichnungen (ohne doppelte Kombinationen):

```
SELECT DISTINCT Lagerhaus_Lager_ID, Bezeichnung FROM Artikel;
```

```
MariaDB [hosakbari_db]> SELECT DISTINCT Lagerhaus_Lager_ID, Bezeichnung FROM Artikel;
+-----+-----+
| Lagerhaus_Lager_ID | Bezeichnung |
+-----+-----+
| 1 | Smartphone-Modell X |
| 2 | Laptop-Modell Y |
| 3 | Tablet-Modell Z |
| 4 | Kühlschrank Eco-Kühl |
| 4 | Waschmaschine SuperClean |
+-----+-----+
```

Zeige alle Kombinationen aus Gewicht und Lagerbestand (ohne doppelte Kombinationen):

```
SELECT DISTINCT Gewicht, Lagerbestand FROM Artikel;
```

```
MariaDB [hosakbari_db]> SELECT DISTINCT Gewicht, Lagerbestand FROM Artikel;
+-----+-----+
| Gewicht | Lagerbestand |
+-----+-----+
| 0.2    | 1500   |
| 1.5    | 500    |
| 0.8    | 800    |
| 50     | 120    |
| 75     | 50     |
+-----+-----+
5 rows in set (0.002 sec)
```

Zeige alle Kombinationen aus Artikelnummer und Bezeichnung (ohne doppelte Kombinationen):

```
SELECT DISTINCT Artikelnummer, Bezeichnung FROM Artikel;
```

```
MariaDB [hosakbari_db]> SELECT DISTINCT Artikelnummer, Bezeichnung FROM Artikel;
+-----+-----+
| Artikelnummer | Bezeichnung |
+-----+-----+
| 101 | Smartphone-Modell X |
| 102 | Laptop-Modell Y |
| 103 | Tablet-Modell Z |
| 104 | Kühlschrank Eco-Kühl |
| 105 | Waschmaschine SuperClean |
+-----+-----+
```

Beispiele mit DISTINCT und Bedingungen:

Es wird eine Abfrage erstellt, die alle eindeutigen Artikelnummern von Artikeln anzeigen, deren Gewicht größer als 50 ist.

```
SELECT DISTINCT Artikelnummer FROM Artikel WHERE Gewicht > 50;
```

Eine weitere Abfrage listet alle unterschiedlichen Bezeichnungen von Artikeln auf, die im Lagerhaus mit der ID 4 gelagert sind.

```
SELECT DISTINCT Bezeichnung FROM Artikel WHERE Lagerhaus_Lager_ID = 4;
```

Die letzte Abfrage zeigt alle verschiedenen Gewichte von Artikeln, deren Lagerbestand unter 500 liegt.

```
SELECT DISTINCT Gewicht FROM Artikel WHERE Lagerbestand < 500;
```

Zeige alle unterschiedlichen Artikelnummern von Artikeln mit einem Gewicht größer als 50 kg:

```
SELECT DISTINCT Artikelnummer FROM Artikel WHERE Gewicht > 50;
```

```
MariaDB [hosakbari_db]> SELECT DISTINCT Artikelnummer FROM Artikel WHERE Gewicht > 50
+-----+
| Artikelnummer |
+-----+
| 105 |
+-----+
```

Zeige alle unterschiedlichen Bezeichnungen von Artikeln, die in Lagerhaus 4 gelagert sind:

```
SELECT DISTINCT Bezeichnung FROM Artikel WHERE Lagerhaus_Lager_ID = 4;
```

```
MariaDB [hosakbari_db]> SELECT DISTINCT Bezeichnung FROM Artikel WHERE Lagerhaus_Lager_ID = 4
+-----+
| Bezeichnung |
+-----+
| Kühlschrank Eco-Kühl |
| Waschmaschine SuperClean |
+-----+
```

Zeige alle unterschiedlichen Gewichtswerte von Artikeln mit einem Lagerbestand kleiner als 500:

```
SELECT DISTINCT Gewicht FROM Artikel WHERE Lagerbestand < 500;
```

Gewicht
50
75

Kombination von DISTINCT mit ORDER BY:

Es wird eine Abfrage erstellt, die alle eindeutigen Lagerhaus-IDs aus der Tabelle Artikel auflistet, absteigend sortiert nach der Lagerhaus-ID.

SELECT DISTINCT Lagerhaus_Lager_ID FROM Artikel ORDER BY Lagerhaus_Lager_ID DESC;

Eine weitere Abfrage zeigt alle verschiedenen Artikelnummern aus der Tabelle Artikel, aufsteigend sortiert nach dem Gewicht.

SELECT DISTINCT Artikelnummer FROM Artikel ORDER BY Gewicht ASC;

Die letzte Abfrage listet alle unterschiedlichen Kombinationen aus Gewicht und Bezeichnung aus der Tabelle Artikel auf, absteigend sortiert nach dem Gewicht.

SELECT DISTINCT Gewicht, Bezeichnung FROM Artikel ORDER BY Gewicht DESC;

Zeige alle unterschiedlichen Lagerhaus-IDs, sortiert absteigend:

SELECT DISTINCT Lagerhaus_Lager_ID FROM Artikel ORDER BY Lagerhaus_Lager_ID DESC;

Lagerhaus_Lager_ID
4
3
2
1

Zeige alle unterschiedlichen Artikelnummern, sortiert nach Gewicht aufsteigend:

SELECT DISTINCT Artikelnummer FROM Artikel ORDER BY Gewicht ASC;

Artikelnummer
101
103
102
104
105

Zeige alle Kombinationen aus Gewicht und Bezeichnung, sortiert nach Gewicht absteigend:

SELECT DISTINCT Gewicht, Bezeichnung FROM Artikel ORDER BY Gewicht DESC;

Gewicht	Bezeichnung
75	Waschmaschine SuperClean
50	Kühlschrank Eco-Kühl
1.5	Laptop-Modell Y
0.8	Tablet-Modell Z
0.2	Smartphone-Modell X

Beispiele mit DISTINCT und Aggregationen:

Es wird eine Abfrage erstellt, die alle eindeutigen Lagerhaus-IDs aus der Tabelle Artikel auflistet, wobei nur Artikel mit einem Lagerbestand von mehr als 100 berücksichtigt werden.

SELECT DISTINCT Lagerhaus_Lager_ID FROM Artikel WHERE Lagerbestand > 100;

Eine weitere Abfrage zeigt alle verschiedenen Kombinationen aus Bezeichnung und Gewicht aus der Tabelle Artikel, wobei nur Artikel mit einem Gewicht von weniger als 10 angezeigt werden.

```
SELECT DISTINCT Bezeichnung, Gewicht FROM Artikel WHERE Gewicht < 10;
```

Die letzte Abfrage listet alle unterschiedlichen Kombinationen aus Lagerbestand und Lagerhaus-ID aus der Tabelle Artikel auf, aufsteigend sortiert nach dem Lagerbestand.

```
SELECT DISTINCT Lagerbestand, Lagerhaus_Lager_ID FROM Artikel ORDER BY Lagerbestand;
```

Zeige alle unterschiedlichen Lagerhaus-IDs, in denen Artikel mit einem Lagerbestand größer als 100 gelagert sind:

```
SELECT DISTINCT Lagerhaus_Lager_ID FROM Artikel WHERE Lagerbestand > 100;
```

```
MariaDB [hosakbari_db]> SELECT DISTINCT Lagerhaus_Lager_ID FROM Artikel WHERE Lagerbestand > 100;
+-----+
| Lagerhaus_Lager_ID |
+-----+
| 1 |
| 2 |
| 3 |
| 4 |
+-----+
```

Zeige alle unterschiedlichen Kombinationen aus Bezeichnung und Gewicht, bei denen das Gewicht kleiner als 10 ist:

```
SELECT DISTINCT Bezeichnung, Gewicht FROM Artikel WHERE Gewicht < 10;
```

```
MariaDB [hosakbari_db]> SELECT DISTINCT Bezeichnung, Gewicht FROM Artikel WHERE Gewicht < 10;
+-----+-----+
| Bezeichnung | Gewicht |
+-----+-----+
| Smartphone-Modell X | 0.2 |
| Laptop-Modell Y | 1.5 |
| Tablet-Modell Z | 0.8 |
+-----+-----+
```

Zeige alle unterschiedlichen Kombinationen aus Lagerbestand und Lagerhaus-IDs, sortiert nach Lagerbestand:

```
SELECT DISTINCT Lagerbestand, Lagerhaus_Lager_ID FROM Artikel ORDER BY Lagerbestand;
```

```
MariaDB [hosakbari_db]> SELECT DISTINCT Lagerbestand, Lagerhaus_Lager_ID FROM Artikel ORDER BY Lagerbestand;
+-----+-----+
| Lagerbestand | Lagerhaus_Lager_ID |
+-----+-----+
| 50 | 4 |
| 120 | 4 |
| 500 | 2 |
| 800 | 3 |
| 1500 | 1 |
+-----+-----+
```

Führen Sie folgende SQL-Operationen (4 Beispiele pro Operation) mit Ihren Testdaten aus: UPDATE, ALTER, DELETE; Relationen mit Nullmarken (4 Beispiele); Problematik „Logistik in 20 Jahren“

Als erstes fügen wir NULL-Werte in verschiedene Tabellen ein, um die nachfolgenden Abfragen sinnvoll durchführen zu können

Fahrzeug davor und danach:

Es wird ein neuer Datensatz in die Tabelle Fahrzeug eingefügt, bei dem ein Fahrzeug mit der ID 5 vom Typ „Autonomer LKW“ und einer Kapazität von 7000 Kilogramm sowie einem Batteriestatus von 90 hinzugefügt wird, jedoch ohne einer Lieferung zugeordnet zu sein.

INSERT INTO Fahrzeug (Fahrzeug_ID, Typ, Kapazität, Batteriestatus, Lieferung_Lieferungsnummer) VALUES (5, 'Autonomer LKW', 7000.0, 90, NULL);

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
+-----+-----+-----+-----+-----+
| Fahrzeug_ID | Typ | Kapazität | Batteriestatus | Lieferung_Lieferungsnummer |
+-----+-----+-----+-----+-----+
| 1 | Elektro-LKW | 5000 | 85 | 1 |
| 2 | Drohne | 50 | 100 | 2 |
| 3 | Elektro-Van | 1000 | 75 | 3 |
| 4 | Hybrid-LKW | 8000 | 60 | NULL |
+-----+-----+-----+-----+-----+
4 rows in set (0,001 sec)

MariaDB [hosakbari_db]> INSERT INTO Fahrzeug (Fahrzeug_ID, Typ, Kapazität, Batteriestatus, Lieferung_Lieferungsnummer)
    -> VALUES (5, 'Autonomer LKW', 7000.0, 90, NULL);
Query OK, 1 row affected (0,021 sec)

MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
+-----+-----+-----+-----+-----+
| Fahrzeug_ID | Typ | Kapazität | Batteriestatus | Lieferung_Lieferungsnummer |
+-----+-----+-----+-----+-----+
| 1 | Elektro-LKW | 5000 | 85 | 1 |
| 2 | Drohne | 50 | 100 | 2 |
| 3 | Elektro-Van | 1000 | 75 | 3 |
| 4 | Hybrid-LKW | 8000 | 60 | NULL |
| 5 | Autonomer LKW | 7000 | 90 | NULL |
+-----+-----+-----+-----+-----+
5 rows in set (0,001 sec)
```

Darüber hinaus werden in die Tabelle Fahrzeug drei weitere Fahrzeuge eingefügt: ein Wasserstoff-LKW mit der ID 6, einer Kapazität von 6500 Kilogramm und einem Batteriestatus von 80, ein Elektro-Scooter mit der ID 7, einer Kapazität von 120 Kilogramm und einem Batteriestatus von 95 sowie eine Hybrid-Drohne mit der ID 8, einer Kapazität von 75 Kilogramm und einem Batteriestatus von 60. Keines dieser Fahrzeuge wird einer Lieferung zugewiesen.

INSERT INTO Fahrzeug (Fahrzeug_ID, Typ, Kapazität, Batteriestatus, Lieferung_Lieferungsnummer)

VALUES

(6, 'Wasserstoff-LKW', 6500.0, 80, NULL),
(7, 'Elektro-Scooter', 120.0, 95, NULL),
(8, 'Hybrid-Drohne', 75.0, 60, NULL);

```
MariaDB [hosakbari_db]> INSERT INTO Fahrzeug (Fahrzeug_ID, Typ, Kapazität, Batteriestatus, Lieferung_Lieferungsnummer)
    -> VALUES
(6, 'Wasserstoff-LKW', 6500.0, 80, NULL),
(7, 'Elektro-Scooter', 120.0, 95, NULL),
(8, 'Hybrid-Drohne', 75.0, 60, NULL);
Query OK, 3 rows affected (0,025 sec)
Records: 3  Duplicates: 0  Warnings: 0

MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'Fahrzeug' at line 1
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
+-----+-----+-----+-----+-----+
| Fahrzeug_ID | Typ | Kapazität | Batteriestatus | Lieferung_Lieferungsnummer |
+-----+-----+-----+-----+-----+
| 1 | Elektro-LKW | 5000 | 85 | 1 |
| 2 | Drohne | 50 | 100 | 2 |
| 3 | Elektro-Van | 1000 | 75 | 3 |
| 4 | Hybrid-LKW | 8000 | 60 | NULL |
| 5 | Autonomer LKW | 7000 | 90 | NULL |
| 6 | Wasserstoff-LKW | 6500 | 80 | NULL |
| 7 | Elektro-Scooter | 120 | 95 | NULL |
| 8 | Hybrid-Drohne | 75 | 60 | NULL |
+-----+-----+-----+-----+-----+
8 rows in set (0,001 sec)
```

Artikel davor und danach:

Ein neuer Artikel mit der Artikelnummer 106, der Bezeichnung „Drucker Modell T100“, einem Gewicht von 5 Kilogramm und einem Lagerbestand von 100 wird der Tabelle Artikel hinzugefügt, jedoch ohne einem bestimmten Lagerhaus zugeordnet zu sein.

```
INSERT INTO Artikel (Artikelnummer, Bezeichnung, Gewicht, Lagerbestand, Lagerhaus_Lager_ID) VALUES (106, 'Drucker Modell T100', 5.0, 100, NULL);
```

```
MariaDB [hosakbari_db]> SELECT * FROM Artikel;
+-----+-----+-----+-----+-----+
| Artikelnummer | Bezeichnung | Gewicht | Lagerbestand | Lagerhaus_Lager_ID |
+-----+-----+-----+-----+-----+
| 101 | Smartphone-Modell X | 0.2 | 1500 | 1 |
| 102 | Laptop-Modell Y | 1.5 | 500 | 2 |
| 103 | Tablet-Modell Z | 0.8 | 800 | 3 |
| 104 | Kühlschrank Eco-Kühl | 50 | 120 | 4 |
| 105 | Waschmaschine SuperClean | 75 | 50 | 4 |
+-----+-----+-----+-----+-----+
5 rows in set (0,001 sec)

MariaDB [hosakbari_db]> INSERT INTO Artikel (Artikelnummer, Bezeichnung, Gewicht, Lagerbestand, Lagerhaus_Lager_ID)
    -> VALUES (106, 'Drucker Modell T100', 5.0, 100, NULL);
Query OK, 1 row affected (0,004 sec)

MariaDB [hosakbari_db]> INSERT INTO Artikel (Artikelnummer, Bezeichnung, Gewicht, Lagerbestand, Lagerhaus_Lager_ID)
    -> ^C
MariaDB [hosakbari_db]> SELECT * FROM Artikel;
+-----+-----+-----+-----+-----+
| Artikelnummer | Bezeichnung | Gewicht | Lagerbestand | Lagerhaus_Lager_ID |
+-----+-----+-----+-----+-----+
| 101 | Smartphone-Modell X | 0.2 | 1500 | 1 |
| 102 | Laptop-Modell Y | 1.5 | 500 | 2 |
| 103 | Tablet-Modell Z | 0.8 | 800 | 3 |
| 104 | Kühlschrank Eco-Kühl | 50 | 120 | 4 |
| 105 | Waschmaschine SuperClean | 75 | 50 | 4 |
| 106 | Drucker Modell T100 | 5 | 100 | NULL |
+-----+-----+-----+-----+-----+
6 rows in set (0,001 sec)
```

Weitere Artikel werden in die Tabelle Artikel eingefügt: ein 3D-Drucker HighTech mit der Artikelnummer 107, einem Gewicht von 10 Kilogramm und einem Lagerbestand von 200, eine Solar-Powerbank mit der Artikelnummer 108, einem Gewicht von 0,5 Kilogramm und einem Lagerbestand von 300 sowie eine Smartwatch Pro mit der Artikelnummer 109, einem Gewicht von 0,3 Kilogramm und einem Lagerbestand von 500. Keiner dieser Artikel ist einem Lagerhaus zugewiesen.

```
INSERT INTO Artikel (Artikelnummer, Bezeichnung, Gewicht, Lagerbestand, Lagerhaus_Lager_ID)
VALUES
(107, '3D-Drucker HighTech', 10.0, 200, NULL),
(108, 'Solar-Powerbank', 0.5, 300, NULL),
(109, 'Smartwatch Pro', 0.3, 500, NULL);
```

```
MariaDB [hosakbari_db]> INSERT INTO Artikel (Artikelnummer, Bezeichnung, Gewicht, Lagerbestand, Lagerhaus_Lager_ID)
    -> VALUES
    -> (107, '3D-Drucker HighTech', 10.0, 200, NULL),
9, 'Smartwatch P' -> (108, 'Solar-Powerbank', 0.5, 300, NULL),
    -> (109, 'Smartwatch Pro', 0.3, 500, NULL);
Query OK, 3 rows affected (0,002 sec)
Records: 3  Duplicates: 0  Warnings: 0

MariaDB [hosakbari_db]> SELECT * FROM Artikel;
+-----+-----+-----+-----+-----+
| Artikelnummer | Bezeichnung | Gewicht | Lagerbestand | Lagerhaus_Lager_ID |
+-----+-----+-----+-----+-----+
| 101 | Smartphone-Modell X | 0.2 | 1500 | 1 |
| 102 | Laptop-Modell Y | 1.5 | 500 | 2 |
| 103 | Tablet-Modell Z | 0.8 | 800 | 3 |
| 104 | Kühlschrank Eco-Kühl | 50 | 120 | 4 |
| 105 | Waschmaschine SuperClean | 75 | 50 | 4 |
| 106 | Drucker Modell T100 | 5 | 100 | NULL |
| 107 | 3D-Drucker HighTech | 10 | 200 | NULL |
| 108 | Solar-Powerbank | 0.5 | 300 | NULL |
| 109 | Smartwatch Pro | 0.3 | 500 | NULL |
+-----+-----+-----+-----+-----+
9 rows in set (0,003 sec)
```

Lieferung davor und danach:

In die Tabelle Lieferung wird ein neuer Eintrag mit der Lieferungsnummer 5, ohne definierten Zielort, einem Lieferzeitpunkt am 20. Juni 2044 und dem Status „in Bearbeitung“ eingefügt.

```
INSERT INTO Lieferung (Lieferungsnummer, Zielort, Lieferzeitpunkt, Status) VALUES (5, NULL, '2044-06-20', 'in Bearbeitung');
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
+-----+-----+-----+-----+
| Lieferungsnummer | Zielort           | Lieferzeitpunkt | Status      |
+-----+-----+-----+-----+
| 1   | Berlin, Alexanderplatz | 2044-06-15    | unterwegs   |
| 2   | Hamburg, Hafenstraße | 2044-06-16    | geliefert   |
| 3   | München, Marienplatz | 2044-06-17    | verspätet   |
| 4   | Frankfurt, Zeilstraße | 2044-06-18    | geplant    |
+-----+-----+-----+-----+
4 rows in set (0,001 sec)

MariaDB [hosakbari_db]> INSERT INTO Lieferung (Lieferungsnummer, Zielort, Lieferzeitpunkt, Status)
    -> VALUES (5, NULL, '2044-06-20', 'in Bearbeitung');
Query OK, 1 row affected (0,002 sec)

MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
+-----+-----+-----+-----+
| Lieferungsnummer | Zielort           | Lieferzeitpunkt | Status      |
+-----+-----+-----+-----+
| 1   | Berlin, Alexanderplatz | 2044-06-15    | unterwegs   |
| 2   | Hamburg, Hafenstraße | 2044-06-16    | geliefert   |
| 3   | München, Marienplatz | 2044-06-17    | verspätet   |
| 4   | Frankfurt, Zeilstraße | 2044-06-18    | geplant    |
| 5   | NULL                 | 2044-06-20    | in Bearbeitung |
+-----+-----+-----+-----+
5 rows in set (0,001 sec)
```

Zudem werden drei weitere Lieferungen hinzugefügt: eine Lieferung mit der Nummer 6, ohne definierten Zielort, einem Lieferzeitpunkt am 21. Juni 2044 und dem Status „bereit zur Abholung“, eine Lieferung mit der Nummer 7, ohne Zielort, einem Lieferzeitpunkt am 22. Juni 2044 und dem Status „verzögert“ sowie eine Lieferung mit der Nummer 8, ohne Zielort, einem Lieferzeitpunkt am 23. Juni 2044 und dem Status „unterwegs“.

```
INSERT INTO Lieferung (Lieferungsnummer, Zielort, Lieferzeitpunkt, Status)
```

```
VALUES
```

```
(6, NULL, '2044-06-21', 'bereit zur Abholung'),
(7, NULL, '2044-06-22', 'verzögert'),
(8, NULL, '2044-06-23', 'unterwegs');
```

```
MariaDB [hosakbari_db]> INSERT INTO Lieferung (Lieferungsnummer, Zielort, Lieferzeitpunkt, Status)
→ VALUES
→ (6, NULL, '2044-06-21', 'bereit zur Abholung'),
→ (7, NULL, '2044-06-22', 'verzögert'),
→ (8, NULL, '2044-06-23', 'unterwegs');
Query OK, 3 rows affected (0,002 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung
Lieferung.Lieferungsnummer  Lieferung.Lieferzeitpunkt  Lieferung.Zielort      Lieferungsnummer
Lieferung.Lieferungsnummer  Lieferung.Status        Lieferung_Lieferungsnummer Lieferzeitpunkt
MariaDB [hosakbari_db]> SELECT * FROM Liefer
Lieferung.Lieferungsnummer  Lieferung.Lieferzeitpunkt  Lieferung.Zielort      Lieferungsnummer
Lieferung.Lieferungsnummer  Lieferung.Status        Lieferung_Lieferungsnummer Lieferzeitpunkt
MariaDB [hosakbari_db]> SELECT * FROM Liefer
Lieferung.Lieferungsnummer  Lieferung.Lieferzeitpunkt  Lieferung.Zielort      Lieferungsnummer
Lieferung.Lieferungsnummer  Lieferung.Status        Lieferung_Lieferungsnummer Lieferzeitpunkt
MariaDB [hosakbari_db]> SELECT * FROM Liefer;
+-----+-----+-----+-----+
| Lieferungsnummer | Zielort          | Lieferzeitpunkt | Status       |
+-----+-----+-----+-----+
| 1   | Berlin, Alexanderplatz | 2044-06-15    | unterwegs    |
| 2   | Hamburg, Hafenstraße | 2044-06-16    | geliefert    |
| 3   | München, Marienplatz | 2044-06-17    | verspätet    |
| 4   | Frankfurt, Zeilstraße | 2044-06-18    | geplant      |
| 5   | NULL                 | 2044-06-20    | in Bearbeitung |
| 6   | NULL                 | 2044-06-21    | bereit zur Abholung |
| 7   | NULL                 | 2044-06-22    | verzögert    |
| 8   | NULL                 | 2044-06-23    | unterwegs    |
+-----+-----+-----+-----+
8 rows in set (0,001 sec)
```

Lagerhaus davor und danach:

In die Tabelle Lagerhaus wird ein neues Lager mit der ID 5, dem Standort „Leipzig, Zentrallager“, einer Kapazität von 40000 und ohne definierte Energiequelle eingefügt.

```
INSERT INTO Lagerhaus (Lager_ID, Standort, Kapazität, Energiequelle) VALUES (5, 'Leipzig, Zentrallager', 40000, NULL);
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
+-----+-----+-----+-----+
| Lager_ID | Standort | Kapazität | Energiequelle |
+-----+-----+-----+-----+
| 1 | Berlin, Lagerstraße 12 | 50000 | Solar |
| 2 | Hamburg, Hafenlager 23 | 75000 | Wind |
| 3 | München, Südallee 45 | 30000 | Solar |
| 4 | Frankfurt, Westhafen 8 | 100000 | Solar/Wind |
+-----+-----+-----+-----+
4 rows in set (0,001 sec)

MariaDB [hosakbari_db]> INSERT INTO Lagerhaus (Lager_ID, Standort, Kapazität, Energiequelle)
    → VALUES (5, 'Leipzig, Zentrallager', 40000, NULL);
Query OK, 1 row affected (0,002 sec)

MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
+-----+-----+-----+-----+
| Lager_ID | Standort | Kapazität | Energiequelle |
+-----+-----+-----+-----+
| 1 | Berlin, Lagerstraße 12 | 50000 | Solar |
| 2 | Hamburg, Hafenlager 23 | 75000 | Wind |
| 3 | München, Südallee 45 | 30000 | Solar |
| 4 | Frankfurt, Westhafen 8 | 100000 | Solar/Wind |
| 5 | Leipzig, Zentrallager | 40000 | NULL |
+-----+-----+-----+-----+
5 rows in set (0,001 sec)
```

Zusätzlich werden drei weitere Lagerhäuser in die Tabelle Lagerhaus eingefügt: ein Lager mit der ID 6 in Stuttgart am Standort „Hauptlager 1“ mit einer Kapazität von 55000, ein Lager mit der ID 7 in Düsseldorf am Standort „Logistikzentrum“ mit einer Kapazität von 45000 sowie ein Lager mit der ID 8 in Dresden am Standort „Verteilzentrum“ mit einer Kapazität von 60000. Für keines dieser Lagerhäuser wird eine Energiequelle angegeben.

```
INSERT INTO Lagerhaus (Lager_ID, Standort, Kapazität, Energiequelle)
```

```
VALUES
```

```
(6, 'Stuttgart, Hauptlager 1', 55000, NULL),  
(7, 'Düsseldorf, Logistikzentrum', 45000, NULL),  
(8, 'Dresden, Verteilzentrum', 60000, NULL);
```

```
MariaDB [hosakbari_db]> INSERT INTO Lagerhaus (Lager_ID, Standort, Kapazität, Energiequelle)  
→ VALUES  
→ (6, 'Stuttgart, Hauptlager 1', 55000, NULL),  
→ (7, 'Düsseldorf, Logistikzentrum', 45000, NULL),  
→ (8, 'Dresden, Verteilzentrum', 60000, NULL);  
Query OK, 3 rows affected (0,002 sec)  
Records: 3  Duplicates: 0  Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
```

Lager_ID	Standort	Kapazität	Energiequelle
1	Berlin, Lagerstraße 12	50000	Solar
2	Hamburg, Hafenlager 23	75000	Wind
3	München, Südallee 45	30000	Solar
4	Frankfurt, Westhafen 8	100000	Solar/Wind
5	Leipzig, Zentrallager	40000	NULL
6	Stuttgart, Hauptlager 1	55000	NULL
7	Düsseldorf, Logistikzentrum	45000	NULL
8	Dresden, Verteilzentrum	60000	NULL

```
8 rows in set (0,001 sec)
```

Relation_Lieferung_Artikel davor und danach:

Schließlich wird in die Tabelle Relation_Lieferung_Artikel ein Eintrag erstellt, der einen Artikel mit der Artikelnummer 106 einer Lieferung zuweist, allerdings ohne eine Lieferungsnummer anzugeben.

```
INSERT INTO Relation_Lieferung_Artikel (Lieferungsnummer, Artikelnummer) VALUES (NULL, 106);
```

```
MariaDB [hosakbari_db]> SELECT * FROM Relation_Lieferung_Artikel;
```

Lieferungsnummer	Artikelnummer
1	101
3	101
1	102
4	102
2	103
4	103
2	104
3	105

```
8 rows in set (0,001 sec)
```

```
MariaDB [hosakbari_db]> INSERT INTO Relation_Lieferung_Artikel (Lieferungsnummer, Artikelnummer)  
    → VALUES (NULL, 106);
```

```
ERROR 1048 (23000): Column 'Lieferungsnummer' cannot be null
```

```
MariaDB [hosakbari_db]> SELECT * FROM Relation_Lieferung_Artikel;
```

Lieferungsnummer	Artikelnummer
1	101
3	101
1	102
4	102
2	103
4	103
2	104
3	105

```
8 rows in set (0,001 sec)
```

```
MariaDB [hosakbari_db]> INSERT INTO Relation_Lieferung_Artikel (Lieferungsnummer, Artikelnummer)  
    → VALUES (NULL, 106);
```

```
ERROR 1048 (23000): Column 'Lieferungsnummer' cannot be null
```

```
MariaDB [hosakbari_db]> SELECT * FROM Relation_Lieferung_Artikel;
```

Lieferungsnummer	Artikelnummer
1	101
3	101
1	102
4	102
2	103
4	103
2	104
3	105

```
8 rows in set (0,001 sec)
```

UPDATE-Operationen für Lieferung

UPDATE-Befehl kommt in SQL zum Einsatz, um Daten die bereits vorhanden sind, in einer Tabelle zu ändern. Die Syntax hierzu ist nach dem Schema aufgebaut:

UPDATE Tabellenname

SET Spalte1 = Wert1, Spalte2 = Wert2, ...

WHERE Bedingung;

Bei den Befehlen unten wird in der Tabelle Lieferung der Status in Bearbeitung gesetzt, wenn Zielort nicht angegeben ist. Der Lieferzeitpunkt wird auf den 1. Juli 2044 gesetzt, falls er fehlt. Falls der Zielort fehlt, wird es durch Unbekannt ersetzt. Falls der Lieferzeitpunkt nach dem 20. Juni 2044 liegt, wird die Lieferung als verspätet markiert.

Es wird ein Update auf die Tabelle Lieferung durchgeführt, um den Status aller Lieferungen auf „in Bearbeitung“ zu setzen, bei denen der Zielort nicht angegeben ist.

```
UPDATE Lieferung SET Status = 'in Bearbeitung' WHERE Zielort IS NULL;
```

Ein weiteres Update wird ausgeführt, um den Lieferzeitpunkt für alle Lieferungen, bei denen bisher kein Lieferzeitpunkt definiert wurde, auf den 1. Juli 2044 zu setzen.

```
UPDATE Lieferung SET Lieferzeitpunkt = '2044-07-01' WHERE Lieferzeitpunkt IS NULL;
```

Zusätzlich wird in allen Datensätzen, in denen der Zielort nicht angegeben ist, der Zielort mit „Unbekannt“ gefüllt.

```
UPDATE Lieferung SET Zielort = 'Unbekannt' WHERE Zielort IS NULL;
```

Abschließend wird ein Update vorgenommen, um den Status aller Lieferungen, deren Lieferzeitpunkt nach dem 20. Juni 2044 liegt, auf „Verspätet“ zu ändern.

```
UPDATE Lieferung SET Status = 'Verspätet' WHERE Lieferzeitpunkt > '2044-06-20';
```

```
MariaDB [hosakbari_db]> UPDATE Lieferung SET Status = 'in Bearbeitung' WHERE Zielort IS NULL;
Query OK, 3 rows affected (0,002 sec)
Rows matched: 4 Changed: 3 Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status
1	Berlin, Alexanderplatz	2044-06-15	unterwegs
2	Hamburg, Hafenstraße	2044-06-16	geliefert
3	München, Marienplatz	2044-06-17	verspätet
4	Frankfurt, Zeilstraße	2044-06-18	geplant
5	NULL	2044-06-20	in Bearbeitung
6	NULL	2044-06-21	in Bearbeitung
7	NULL	2044-06-22	in Bearbeitung
8	NULL	2044-06-23	in Bearbeitung

8 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> UPDATE Lieferung SET Lieferzeitpunkt = '2044-07-01' WHERE Lieferzeitpunkt IS NULL
Query OK, 0 rows affected (0,001 sec)
Rows matched: 0 Changed: 0 Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status
1	Berlin, Alexanderplatz	2044-06-15	unterwegs
2	Hamburg, Hafenstraße	2044-06-16	geliefert
3	München, Marienplatz	2044-06-17	verspätet
4	Frankfurt, Zeilstraße	2044-06-18	geplant
5	NULL	2044-06-20	in Bearbeitung
6	NULL	2044-06-21	in Bearbeitung
7	NULL	2044-06-22	in Bearbeitung
8	NULL	2044-06-23	in Bearbeitung

8 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> UPDATE Lieferung SET Zielort = 'Unbekannt' WHERE Zielort IS NULL
Query OK, 4 rows affected (0,002 sec)
Rows matched: 4 Changed: 4 Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status
1	Berlin, Alexanderplatz	2044-06-15	unterwegs
2	Hamburg, Hafenstraße	2044-06-16	geliefert
3	München, Marienplatz	2044-06-17	verspätet
4	Frankfurt, Zeilstraße	2044-06-18	geplant
5	Unbekannt	2044-06-20	in Bearbeitung
6	Unbekannt	2044-06-21	in Bearbeitung
7	Unbekannt	2044-06-22	in Bearbeitung
8	Unbekannt	2044-06-23	in Bearbeitung

8 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> UPDATE Lieferung SET Status = 'Verspätet' WHERE Lieferzeitpunkt > '2044-06-20'
Query OK, 3 rows affected (0,002 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status
1	Berlin, Alexanderplatz	2044-06-15	unterwegs
2	Hamburg, Hafenstraße	2044-06-16	geliefert
3	München, Marienplatz	2044-06-17	verspätet
4	Frankfurt, Zeilstraße	2044-06-18	geplant
5	Unbekannt	2044-06-20	in Bearbeitung
6	Unbekannt	2044-06-21	Verspätet
7	Unbekannt	2044-06-22	Verspätet
8	Unbekannt	2044-06-23	Verspätet

8 rows in set (0,001 sec)

UPDATE-Operation für Fahrzeug

Hier unten wird bei der Tabelle Fahrzeug der Batteriestatus auf 100 gesetzt, wenn das Fahrzeug keiner Lieferung zugeordnet ist. Wenn die Kapazität unter 5000 liegt, wird sie um 10 Prozent erhöht. Der Typ wird auf Hybrid-Fahrzeug geändert, wenn der Batteriestatus unter 50 liegt. Zuletzt werden Fahrzeuge ohne Lieferungsnummer, die Nummer 3 hinzugefügt.

Es wird ein Update auf die Tabelle Fahrzeug durchgeführt, um den Batteriestatus aller Fahrzeuge auf 100 zu setzen, bei denen keine Lieferungsnummer zugewiesen ist.

```
UPDATE Fahrzeug SET Batteriestatus = 100 WHERE Lieferung_Lieferungsnummer IS NULL;
```

Ein weiteres Update wird ausgeführt, um die Kapazität aller Fahrzeuge, deren Kapazität unter 5000 liegt, um 10 % zu erhöhen.

```
UPDATE Fahrzeug SET Kapazität = Kapazität * 1.1 WHERE Kapazität < 5000;
```

Darüber hinaus wird der Typ aller Fahrzeuge mit einem Batteriestatus unter 50 auf „Hybrid-Fahrzeug“ geändert.

```
UPDATE Fahrzeug SET Typ = 'Hybrid-Fahrzeug' WHERE Batteriestatus < 50;
```

Abschließend wird ein Update vorgenommen, um allen Fahrzeugen, die bisher keiner Lieferung zugewiesen wurden, die Lieferungsnummer 3 zuzuweisen.

```
UPDATE Fahrzeug SET Lieferung_Lieferungsnummer = 3 WHERE Lieferung_Lieferungsnummer IS NULL;
```

```
MariaDB [hosakbari_db]> UPDATE Fahrzeug SET Batteriestatus = 100 WHERE Lieferung_Lieferungsnummer IS NULL
Query OK, 5 rows affected (0,002 sec)
Rows matched: 5  Changed: 5  Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
```

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer
1	Elektro-LKW	5000	85	1
2	Drohne	50	100	2
3	Elektro-Van	1000	75	3
4	Hybrid-LKW	8000	100	NULL
5	Autonomer LKW	7000	100	NULL
6	Wasserstoff-LKW	6500	100	NULL
7	Elektro-Scooter	120	100	NULL
8	Hybrid-Drohne	75	100	NULL

8 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> UPDATE Fahrzeug SET Kapazität = Kapazität * 1.1 WHERE Kapazität < 5000
Query OK, 4 rows affected (0,003 sec)
Rows matched: 4 Changed: 4 Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
```

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer
1	Elektro-LKW	5000	85	1
2	Drohne	55	100	2
3	Elektro-Van	1100	75	3
4	Hybrid-LKW	8000	100	NULL
5	Autonomer LKW	7000	100	NULL
6	Wasserstoff-LKW	6500	100	NULL
7	Elektro-Scooter	132	100	NULL
8	Hybrid-Drohne	82.5	100	NULL

```
8 rows in set (0,001 sec)
```

```
MariaDB [hosakbari_db]> UPDATE Fahrzeug SET Typ = 'Hybrid-Fahrzeug' WHERE Batteriestatus < 50
Query OK, 0 rows affected (0,001 sec)
Rows matched: 0 Changed: 0 Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
```

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer
1	Elektro-LKW	5000	85	1
2	Drohne	55	100	2
3	Elektro-Van	1100	75	3
4	Hybrid-LKW	8000	100	NULL
5	Autonomer LKW	7000	100	NULL
6	Wasserstoff-LKW	6500	100	NULL
7	Elektro-Scooter	132	100	NULL
8	Hybrid-Drohne	82.5	100	NULL

```
8 rows in set (0,001 sec)
```

```
MariaDB [hosakbari_db]> UPDATE Fahrzeug SET Lieferung_Lieferungsnummer = 3 WHERE Lieferung_Lieferungsnummer IS NULL
Query OK, 5 rows affected (0,003 sec)
Rows matched: 5 Changed: 5 Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
```

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer
1	Elektro-LKW	5000	85	1
2	Drohne	55	100	2
3	Elektro-Van	1100	75	3
4	Hybrid-LKW	8000	100	3
5	Autonomer LKW	7000	100	3
6	Wasserstoff-LKW	6500	100	3
7	Elektro-Scooter	132	100	3
8	Hybrid-Drohne	82.5	100	3

```
8 rows in set (0,001 sec)
```

UPDATE-Operation für Lagerhaus

Hier wird, wenn die Energiequelle nicht angegeben ist, auf Erneuerbar gesetzt. Wenn die Kapazität unter 50.000 liegt, wird sie um 5000 erhöht. Wenn der Standort nicht angegeben ist, wird sie auf Ort Unbekannt gesetzt. Wenn die Kapazität über 90.000 liegt, wird die Energiequelle auf „Solar“ gesetzt.

Ein Update auf die Tabelle Lagerhaus wird durchgeführt, um für alle Einträge, bei denen die Energiequelle nicht angegeben ist, die Energiequelle auf „Erneuerbar“ zu setzen.

```
UPDATE Lagerhaus SET Energiequelle = 'Erneuerbar' WHERE Energiequelle IS NULL;
```

Ein weiteres Update erhöht die Kapazität aller Lagerhäuser mit einer Kapazität unter 50000 um 5000.

```
UPDATE Lagerhaus SET Kapazität = Kapazität + 5000 WHERE Kapazität < 50000;
```

Zusätzlich wird der Standort aller Lagerhäuser, bei denen der Standort nicht angegeben ist, auf „Ort unbekannt“ gesetzt.

```
UPDATE Lagerhaus SET Standort = 'Ort unbekannt' WHERE Standort IS NULL;
```

Abschließend wird die Energiequelle aller Lagerhäuser mit einer Kapazität über 90000 auf „Solar“ geändert.

```
UPDATE Lagerhaus SET Energiequelle = 'Solar' WHERE Kapazität > 90000;
```

```
MariaDB [hosakbari_db]> UPDATE Lagerhaus SET Energiequelle = 'Erneuerbar' WHERE Energiequelle IS NULL  
Query OK, 4 rows affected (0,003 sec)  
Rows matched: 4 Changed: 4 Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
```

Lager_ID	Standort	Kapazität	Energiequelle
1	Berlin, Lagerstraße 12	50000	Solar
2	Hamburg, Hafenlager 23	75000	Wind
3	München, Südallee 45	30000	Solar
4	Frankfurt, Westhafen 8	100000	Solar/Wind
5	Leipzig, Zentrallager	40000	Erneuerbar
6	Stuttgart, Hauptlager 1	55000	Erneuerbar
7	Düsseldorf, Logistikzentrum	45000	Erneuerbar
8	Dresden, Verteilzentrum	60000	Erneuerbar

8 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> UPDATE Lagerhaus SET Kapazität = Kapazität + 5000 WHERE Kapazität < 50000  
Query OK, 3 rows affected (0,002 sec)  
Rows matched: 3 Changed: 3 Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
```

Lager_ID	Standort	Kapazität	Energiequelle
1	Berlin, Lagerstraße 12	50000	Solar
2	Hamburg, Hafenlager 23	75000	Wind
3	München, Südallee 45	35000	Solar
4	Frankfurt, Westhafen 8	100000	Solar/Wind
5	Leipzig, Zentrallager	45000	Erneuerbar
6	Stuttgart, Hauptlager 1	55000	Erneuerbar
7	Düsseldorf, Logistikzentrum	50000	Erneuerbar
8	Dresden, Verteilzentrum	60000	Erneuerbar

8 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> UPDATE Lagerhaus SET Standort = 'Ort unbekannt' WHERE Standort IS NULL;  
Query OK, 0 rows affected (0,001 sec)  
Rows matched: 0 Changed: 0 Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
```

Lager_ID	Standort	Kapazität	Energiequelle
1	Berlin, Lagerstraße 12	50000	Solar
2	Hamburg, Hafenlager 23	75000	Wind
3	München, Südallee 45	35000	Solar
4	Frankfurt, Westhafen 8	100000	Solar/Wind
5	Leipzig, Zentrallager	45000	Erneuerbar
6	Stuttgart, Hauptlager 1	55000	Erneuerbar
7	Düsseldorf, Logistikzentrum	50000	Erneuerbar
8	Dresden, Verteilzentrum	60000	Erneuerbar

8 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> UPDATE Lagerhaus SET Energiequelle = 'Solar' WHERE Kapazität > 90000
Query OK, 1 row affected (0,002 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
```

Lager_ID	Standort	Kapazität	Energiequelle
1	Berlin, Lagerstraße 12	50000	Solar
2	Hamburg, Hafenlager 23	75000	Wind
3	München, Südallee 45	35000	Solar
4	Frankfurt, Westhafen 8	100000	Solar
5	Leipzig, Zentrallager	45000	Erneuerbar
6	Stuttgart, Hauptlager 1	55000	Erneuerbar
7	Düsseldorf, Logistikzentrum	50000	Erneuerbar
8	Dresden, Verteilzentrum	60000	Erneuerbar

8 rows in set (0,001 sec)

UPDATE-Operation für Artikel

Bei den Befehlen unten wird der Lagerbestand um 50 reduziert, wenn er größer als 100 ist. Die Bezeichnung wird auf Neues Produkt gesetzt, wenn die Artikelnummer 108 ist. Artikeln ohne Lagerhaus_Lager_ID wird die ID 1 hinzugewiesen. Wenn das Gewicht unter 5 liegt, wird das Gewicht um 20 Prozent erhöht.

Ein Update auf die Tabelle Artikel wird durchgeführt, um den Lagerbestand aller Artikel, deren Lagerbestand über 100 liegt, um 50 zu reduzieren.

```
UPDATE Artikel SET Lagerbestand = Lagerbestand - 50 WHERE Lagerbestand > 100;
```

Ein weiteres Update ändert die Bezeichnung des Artikels mit der Artikelnummer 108 in „Neues Produkt“.

```
UPDATE Artikel SET Bezeichnung = 'Neues Produkt' WHERE Artikelnummer = 108;
```

Zusätzlich wird die Lagerhaus-ID aller Artikel, bei denen diese nicht angegeben ist, auf 1 gesetzt.

```
UPDATE Artikel SET Lagerhaus_Lager_ID = 1 WHERE Lagerhaus_Lager_ID IS NULL;
```

Abschließend wird das Gewicht aller Artikel, deren Gewicht unter 5 liegt, um 20 % erhöht.

```
UPDATE Artikel SET Gewicht = Gewicht * 1.2 WHERE Gewicht < 5;
```

```
MariaDB [hosakbari_db]> UPDATE Artikel SET Lagerbestand = Lagerbestand - 50 WHERE Lagerbestand > 100
Query OK, 7 rows affected (0,002 sec)
Rows matched: 7  Changed: 7  Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Artikel;
```

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID
101	Smartphone-Modell X	0.2	1450	1
102	Laptop-Modell Y	1.5	450	2
103	Tablet-Modell Z	0.8	750	3
104	Kühlschrank Eco-Kühl	50	70	4
105	Waschmaschine SuperClean	75	50	4
106	Drucker Modell T100	5	100	NULL
107	3D-Drucker HighTech	10	150	NULL
108	Solar-Powerbank	0.5	250	NULL
109	Smartwatch Pro	0.3	450	NULL

9 rows in set (0,001 sec)

```

MariaDB [hosakbari_db]> UPDATE Artikel SET Bezeichnung = 'Neues Produkt' WHERE Artikelnummer = 108
Query OK, 1 row affected (0,002 sec)
Rows matched: 1 Changed: 1 Warnings: 0

MariaDB [hosakbari_db]> SELECT * FROM Artikel;
+-----+-----+-----+-----+-----+
| Artikelnummer | Bezeichnung | Gewicht | Lagerbestand | Lagerhaus_Lager_ID |
+-----+-----+-----+-----+-----+
| 101 | Smartphone-Modell X | 0.2 | 1450 | 1 |
| 102 | Laptop-Modell Y | 1.5 | 450 | 2 |
| 103 | Tablet-Modell Z | 0.8 | 750 | 3 |
| 104 | Kühlschrank Eco-Kühl | 50 | 70 | 4 |
| 105 | Waschmaschine SuperClean | 75 | 50 | 4 |
| 106 | Drucker Modell T100 | 5 | 100 | NULL |
| 107 | 3D-Drucker HighTech | 10 | 150 | NULL |
| 108 | Neues Produkt | 0.5 | 250 | NULL |
| 109 | Smartwatch Pro | 0.3 | 450 | NULL |
+-----+-----+-----+-----+-----+
9 rows in set (0,001 sec)

MariaDB [hosakbari_db]> UPDATE Artikel SET Lagerhaus_Lager_ID = 1 WHERE Lagerhaus_Lager_ID IS NULL;
Query OK, 4 rows affected (0,003 sec)
Rows matched: 4 Changed: 4 Warnings: 0

MariaDB [hosakbari_db]> SELECT * FROM Artikel;
+-----+-----+-----+-----+-----+
| Artikelnummer | Bezeichnung | Gewicht | Lagerbestand | Lagerhaus_Lager_ID |
+-----+-----+-----+-----+-----+
| 101 | Smartphone-Modell X | 0.2 | 1450 | 1 |
| 102 | Laptop-Modell Y | 1.5 | 450 | 2 |
| 103 | Tablet-Modell Z | 0.8 | 750 | 3 |
| 104 | Kühlschrank Eco-Kühl | 50 | 70 | 4 |
| 105 | Waschmaschine SuperClean | 75 | 50 | 4 |
| 106 | Drucker Modell T100 | 5 | 100 | 1 |
| 107 | 3D-Drucker HighTech | 10 | 150 | 1 |
| 108 | Neues Produkt | 0.5 | 250 | 1 |
| 109 | Smartwatch Pro | 0.3 | 450 | 1 |
+-----+-----+-----+-----+-----+
9 rows in set (0,001 sec)

MariaDB [hosakbari_db]> UPDATE Artikel SET Gewicht = Gewicht * 1.2 WHERE Gewicht < 5;
Query OK, 5 rows affected (0,002 sec)
Rows matched: 5 Changed: 5 Warnings: 0

MariaDB [hosakbari_db]> SELECT * FROM Artikel;
+-----+-----+-----+-----+-----+
| Artikelnummer | Bezeichnung | Gewicht | Lagerbestand | Lagerhaus_Lager_ID |
+-----+-----+-----+-----+-----+
| 101 | Smartphone-Modell X | 0.24 | 1450 | 1 |
| 102 | Laptop-Modell Y | 1.8 | 450 | 2 |
| 103 | Tablet-Modell Z | 0.96 | 750 | 3 |
| 104 | Kühlschrank Eco-Kühl | 50 | 70 | 4 |
| 105 | Waschmaschine SuperClean | 75 | 50 | 4 |
| 106 | Drucker Modell T100 | 5 | 100 | 1 |
| 107 | 3D-Drucker HighTech | 10 | 150 | 1 |
| 108 | Neues Produkt | 0.6 | 250 | 1 |
| 109 | Smartwatch Pro | 0.36 | 450 | 1 |
+-----+-----+-----+-----+-----+
9 rows in set (0,001 sec)

```

UPDATE-Operation für Relation_Lieferung_Artikel

Unten werden erstmal Einträgen ohne Lieferungsnummer, die Nummer 2 hinzugewiesen. Danach wird die Artikelnummer von 109 auf 101 geändert. Im Nachhinein wird die Lieferungsnummer von 3 durch 4 ersetzt. Zuletzt wird die Artikelnummer von 102 auf 104 geändert.

Ein Update auf die Tabelle Relation_Lieferung_Artikel wird durchgeführt, um allen Einträgen, bei denen die Lieferungsnummer nicht angegeben ist, den Wert 2 zuzuweisen.

```
UPDATE Relation_Lieferung_Artikel SET Lieferungsnummer = 2 WHERE Lieferungsnummer IS NULL;
```

Ein weiteres Update ändert den Artikel mit der Artikelnummer 109 zu 101.

```
UPDATE Relation_Lieferung_Artikel SET Artikelnummer = 101 WHERE Artikelnummer = 109;
```

Zusätzlich wird die Lieferungsnummer 3 in der Tabelle auf 4 geändert.

```
UPDATE Relation_Lieferung_Artikel SET Lieferungsnummer = 4 WHERE Lieferungsnummer = 3;
```

Abschließend wird die Artikelnummer 102 auf 104 aktualisiert.

```
UPDATE Relation_Lieferung_Artikel SET Artikelnummer = 104 WHERE Artikelnummer = 102;
```

```
MariaDB [hosakbari_db]> UPDATE Relation_Lieferung_Artikel SET Lieferungsnummer = 2 WHERE Lieferungsnummer IS NULL;  
Query OK, 0 rows affected (0,001 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Relation_Lieferung_Artikel;
```

Lieferungsnummer	Artikelnummer
1	101
3	101
1	102
4	102
2	103
4	103
2	104
3	105

```
8 rows in set (0,001 sec)
```

```
MariaDB [hosakbari_db]> UPDATE Relation_Lieferung_Artikel SET Artikelnummer = 101 WHERE Artikelnummer = 109
```

```
Query OK, 0 rows affected (0,001 sec)
```

```
Rows matched: 0 Changed: 0 Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Relation_Lieferung_Artikel;
```

Lieferungsnummer	Artikelnummer
1	101
3	101
1	102
4	102
2	103
4	103
2	104
3	105

```
8 rows in set (0,001 sec)
```

```
MariaDB [hosakbari_db]> UPDATE Relation_Lieferung_Artikel SET Lieferungsnummer = 4 WHERE Lieferungsnummer = 3
```

```
Query OK, 2 rows affected (0,018 sec)
```

```
Rows matched: 2 Changed: 2 Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Relation_Lieferung_Artikel;
```

Lieferungsnummer	Artikelnummer
1	101
4	101
1	102
4	102
2	103
4	103
2	104
4	105

```
8 rows in set (0,001 sec)
```

```
MariaDB [hosakbari_db]> UPDATE Relation_Lieferung_Artikel SET Artikelnummer = 104 WHERE Artikelnummer = 102
```

```
Query OK, 2 rows affected (0,002 sec)
```

```
Rows matched: 2 Changed: 2 Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Relation_Lieferung_Artikel;
```

Lieferungsnummer	Artikelnummer
1	101
4	101
2	103
4	103
1	104
2	104
4	104
4	105

```
8 rows in set (0,001 sec)
```

ALTER TABLE-Befehl wird in SQL verwendet, um die Struktur einer Tabelle zu ändern. Hierdurch können Spalten hinzugefügt, entfernt oder geändert werden. Grundlegende Syntax hierfür lautet:

```
ALTER TABLE Tabellenname  
[ADD | DROP | MODIFY | ALTER] Spaltenname Datentyp [NULL | NOT NULL];
```

Unten wird der Tabelle Lieferung eine neue Spalte Priorität vom Typ INT hinzugefügt, die NULL-Werte enthalten darf. Dann wird eine neue Spalte Transporttyp vom Typ VARCHAR(50) hinzugefügt, die NULL-Werte enthalten darf. Des Weiteren wird die Spalte Zielort so geändert, dass sie jetzt bis zu 500 Zeichen vom Typ VARCHAR speichern kann. Zuletzt wird eine neue Spalte Versandkosten hinzugefügt, die Zahlen mit bis zu 10 Stellen und 2 Nachkommastellen speichern kann und NULL-Werte erlaubt.

Es wird der Tabelle Lieferung eine neue Spalte Priorität hinzugefügt, die Werte vom Typ INT aufnehmen kann und auch NULL-Werte zulässt.

```
ALTER TABLE Lieferung ADD Priorität INT NULL;
```

Außerdem wird der Tabelle eine neue Spalte Transporttyp vom Typ VARCHAR(50) hinzugefügt, die ebenfalls NULL-Werte enthalten kann.

```
ALTER TABLE Lieferung ADD Transporttyp VARCHAR(50) NULL;
```

Die bestehende Spalte Zielort wird so modifiziert, dass sie nun bis zu 500 Zeichen aufnehmen kann.

```
ALTER TABLE Lieferung MODIFY Zielort VARCHAR(500);
```

Schließlich wird der Tabelle eine weitere Spalte Versandkosten hinzugefügt, die Dezimalwerte mit bis zu 10 Stellen, davon 2 Nachkommastellen, speichert und NULL-Werte zulässt.

```
ALTER TABLE Lieferung ADD Versandkosten DECIMAL(10,2) NULL;
```

```
MariaDB [hosakbari_db]> ALTER TABLE Lieferung ADD Priorität INT NULL;  
Query OK, 0 rows affected (0,023 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität
1	Berlin, Alexanderplatz	2044-06-15	unterwegs	NULL
2	Hamburg, Hafenstraße	2044-06-16	geliefert	NULL
3	München, Marienplatz	2044-06-17	verspätet	NULL
4	Frankfurt, Zeilstraße	2044-06-18	geplant	NULL
5	Unbekannt	2044-06-20	in Bearbeitung	NULL
6	Unbekannt	2044-06-21	Verspätet	NULL
7	Unbekannt	2044-06-22	Verspätet	NULL
8	Unbekannt	2044-06-23	Verspätet	NULL

```
8 rows in set (0,002 sec)
```

```
MariaDB [hosakbari_db]> ALTER TABLE Lieferung ADD Transporttyp VARCHAR(50) NULL;  
Query OK, 0 rows affected (0,013 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp
1	Berlin, Alexanderplatz	2044-06-15	unterwegs	NULL	NULL
2	Hamburg, Hafenstraße	2044-06-16	geliefert	NULL	NULL
3	München, Marienplatz	2044-06-17	verspätet	NULL	NULL
4	Frankfurt, Zeilstraße	2044-06-18	geplant	NULL	NULL
5	Unbekannt	2044-06-20	in Bearbeitung	NULL	NULL
6	Unbekannt	2044-06-21	Verspätet	NULL	NULL
7	Unbekannt	2044-06-22	Verspätet	NULL	NULL
8	Unbekannt	2044-06-23	Verspätet	NULL	NULL

```
8 rows in set (0,002 sec)
```

```
MariaDB [hosakbari_db]> ALTER TABLE Lieferung MODIFY Zielort VARCHAR(500);
Query OK, 0 rows affected (0,012 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
+-----+-----+-----+-----+-----+-----+
| Lieferungsnummer | Zielort | Lieferzeitpunkt | Status | Priorität | Transporttyp |
+-----+-----+-----+-----+-----+-----+
| 1 | Berlin, Alexanderplatz | 2044-06-15 | unterwegs | NULL | NULL |
| 2 | Hamburg, Hafenstraße | 2044-06-16 | geliefert | NULL | NULL |
| 3 | München, Marienplatz | 2044-06-17 | verspätet | NULL | NULL |
| 4 | Frankfurt, Zeilstraße | 2044-06-18 | geplant | NULL | NULL |
| 5 | Unbekannt | 2044-06-20 | in Bearbeitung | NULL | NULL |
| 6 | Unbekannt | 2044-06-21 | Verspätet | NULL | NULL |
| 7 | Unbekannt | 2044-06-22 | Verspätet | NULL | NULL |
| 8 | Unbekannt | 2044-06-23 | Verspätet | NULL | NULL |
+-----+-----+-----+-----+-----+-----+
8 rows in set (0,002 sec)

MariaDB [hosakbari_db]> ALTER TABLE Lieferung ADD Versandkosten DECIMAL(10,2) NULL;
Query OK, 0 rows affected (0,014 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
+-----+-----+-----+-----+-----+-----+-----+
| Lieferungsnummer | Zielort | Lieferzeitpunkt | Status | Priorität | Transporttyp | Versandkosten |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Berlin, Alexanderplatz | 2044-06-15 | unterwegs | NULL | NULL | NULL |
| 2 | Hamburg, Hafenstraße | 2044-06-16 | geliefert | NULL | NULL | NULL |
| 3 | München, Marienplatz | 2044-06-17 | verspätet | NULL | NULL | NULL |
| 4 | Frankfurt, Zeilstraße | 2044-06-18 | geplant | NULL | NULL | NULL |
| 5 | Unbekannt | 2044-06-20 | in Bearbeitung | NULL | NULL | NULL |
| 6 | Unbekannt | 2044-06-21 | Verspätet | NULL | NULL | NULL |
| 7 | Unbekannt | 2044-06-22 | Verspätet | NULL | NULL | NULL |
| 8 | Unbekannt | 2044-06-23 | Verspätet | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+
8 rows in set (0,002 sec)
```

ALTER-Operation für Fahrzeug

Ähnliches gilt für die Befehle unten (unterschiede sind nicht so groß):

Der Tabelle Fahrzeug wird eine neue Spalte Wartungsstatus vom Typ VARCHAR(100) hinzugefügt, die auch NULL-Werte enthalten kann.

```
ALTER TABLE Fahrzeug ADD Wartungsstatus VARCHAR(100) NULL;
```

Zusätzlich wird eine weitere Spalte Ladezeit vom Typ INT ergänzt, die ebenfalls NULL-Werte aufnehmen darf.

```
ALTER TABLE Fahrzeug ADD Ladezeit INT NULL;
```

Die bestehende Spalte Typ wird so geändert, dass sie nun bis zu 100 Zeichen lang sein kann.

```
ALTER TABLE Fahrzeug MODIFY Typ VARCHAR(100);
```

Zuletzt wird der Tabelle eine neue Spalte CO2_Einsparung vom Typ DECIMAL(5,2) hinzugefügt, um CO2-Einsparungswerte mit bis zu fünf Stellen, davon zwei Nachkommastellen, zu speichern, wobei NULL-Werte möglich sind.

```
ALTER TABLE Fahrzeug ADD CO2_Einsparung DECIMAL(5,2) NULL;
```

```
MariaDB [hosakbari_db]> ALTER TABLE Fahrzeug ADD Wartungsstatus VARCHAR(100) NULL;
Query OK, 0 rows affected (0,012 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
```

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer	Wartungsstatus
1	Elektro-LKW	5000	85		1 NULL
2	Drohne	55	100		2 NULL
3	Elektro-Van	1100	75		3 NULL
4	Hybrid-LKW	8000	100		3 NULL
5	Autonomer LKW	7000	100		3 NULL
6	Wasserstoff-LKW	6500	100		3 NULL
7	Elektro-Scooter	132	100		3 NULL
8	Hybrid-Drohne	82.5	100		3 NULL

```
8 rows in set (0,002 sec)
```

```

MariaDB [hosakbari_db]> ALTER TABLE Fahrzeug ADD Ladezeit INT NULL;
Query OK, 0 rows affected (0.014 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
+-----+-----+-----+-----+-----+-----+-----+
| Fahrzeug_ID | Typ | Kapazität | Batteriestatus | Lieferung_Lieferungsnummer | Wartungsstatus | Ladezeit |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Elektro-LKW | 5000 | 85 | 1 | NULL | NULL |
| 2 | Drohne | 55 | 100 | 2 | NULL | NULL |
| 3 | Elektro-Van | 1100 | 75 | 3 | NULL | NULL |
| 4 | Hybrid-LKW | 8000 | 100 | 3 | NULL | NULL |
| 5 | Autonomer LKW | 7000 | 100 | 3 | NULL | NULL |
| 6 | Wasserstoff-LKW | 6500 | 100 | 3 | NULL | NULL |
| 7 | Elektro-Scooter | 132 | 100 | 3 | NULL | NULL |
| 8 | Hybrid-Drohne | 82.5 | 100 | 3 | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+
8 rows in set (0.002 sec)

MariaDB [hosakbari_db]> ALTER TABLE Fahrzeug MODIFY Typ VARCHAR(100);
Query OK, 8 rows affected (0.045 sec)
Records: 8  Duplicates: 0  Warnings: 0

MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
+-----+-----+-----+-----+-----+-----+-----+
| Fahrzeug_ID | Typ | Kapazität | Batteriestatus | Lieferung_Lieferungsnummer | Wartungsstatus | Ladezeit |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Elektro-LKW | 5000 | 85 | 1 | NULL | NULL |
| 2 | Drohne | 55 | 100 | 2 | NULL | NULL |
| 3 | Elektro-Van | 1100 | 75 | 3 | NULL | NULL |
| 4 | Hybrid-LKW | 8000 | 100 | 3 | NULL | NULL |
| 5 | Autonomer LKW | 7000 | 100 | 3 | NULL | NULL |
| 6 | Wasserstoff-LKW | 6500 | 100 | 3 | NULL | NULL |
| 7 | Elektro-Scooter | 132 | 100 | 3 | NULL | NULL |
| 8 | Hybrid-Drohne | 82.5 | 100 | 3 | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+
8 rows in set (0.003 sec)

MariaDB [hosakbari_db]> ALTER TABLE Fahrzeug ADD CO2_Einsparung DECIMAL(5,2) NULL;
ERROR 1060 (42S21): Duplicate column name 'CO2_Einsparung'
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Fahrzeug_ID | Typ | Kapazität | Batteriestatus | Lieferung_Lieferungsnummer | Wartungsstatus | Ladezeit | CO2_Einsparung |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Elektro-LKW | 5000 | 85 | 1 | NULL | NULL | NULL |
| 2 | Drohne | 55 | 100 | 2 | NULL | NULL | NULL |
| 3 | Elektro-Van | 1100 | 75 | 3 | NULL | NULL | NULL |
| 4 | Hybrid-LKW | 8000 | 100 | 3 | NULL | NULL | NULL |
| 5 | Autonomer LKW | 7000 | 100 | 3 | NULL | NULL | NULL |
| 6 | Wasserstoff-LKW | 6500 | 100 | 3 | NULL | NULL | NULL |
| 7 | Elektro-Scooter | 132 | 100 | 3 | NULL | NULL | NULL |
| 8 | Hybrid-Drohne | 82.5 | 100 | 3 | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
8 rows in set (0.001 sec)

```

ALTER-Operation für Lagerhaus

Dito für Lagerhaus

Der Tabelle Lagerhaus wird eine neue Spalte Sicherheitsstufe vom Typ INT hinzugefügt, die NULL-Werte enthalten kann.

ALTER TABLE Lagerhaus ADD Sicherheitsstufe INT NULL;

Außerdem wird eine weitere Spalte Kühlkapazität vom Typ INT ergänzt, die ebenfalls NULL-Werte aufnehmen darf.

ALTER TABLE Lagerhaus ADD Kühlkapazität INT NULL;

Die bestehende Spalte Standort wird so geändert, dass sie nun bis zu 300 Zeichen lang sein kann.

ALTER TABLE Lagerhaus MODIFY Standort VARCHAR(300);

Zuletzt wird der Tabelle Lagerhaus eine neue Spalte Drohnenlandeplatz vom Typ BOOLEAN hinzugefügt, die angibt, ob ein Drohnenlandeplatz vorhanden ist, und NULL-Werte zulässt.

ALTER TABLE Lagerhaus ADD Drohnenlandeplatz BOOLEAN NULL;

```

MariaDB [hosakbari_db]> ALTER TABLE Lagerhaus ADD Sicherheitsstufe INT NULL;
Query OK, 0 rows affected (0,022 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
+-----+-----+-----+-----+
| Lager_ID | Standort | Kapazität | Energiequelle | Sicherheitsstufe |
+-----+-----+-----+-----+
| 1 | Berlin, Lagerstraße 12 | 50000 | Solar | NULL |
| 2 | Hamburg, Hafenlager 23 | 75000 | Wind | NULL |
| 3 | München, Südallee 45 | 35000 | Solar | NULL |
| 4 | Frankfurt, Westhafen 8 | 100000 | Solar | NULL |
| 5 | Leipzig, Zentrallager | 45000 | Erneuerbar | NULL |
| 6 | Stuttgart, Hauptlager 1 | 55000 | Erneuerbar | NULL |
| 7 | Düsseldorf, Logistikzentrum | 50000 | Erneuerbar | NULL |
| 8 | Dresden, Verteilzentrum | 60000 | Erneuerbar | NULL |
+-----+-----+-----+-----+
8 rows in set (0,002 sec)

MariaDB [hosakbari_db]> ALTER TABLE Lagerhaus ADD Kühlkapazität INT NULL;
Query OK, 0 rows affected (0,014 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
+-----+-----+-----+-----+-----+
| Lager_ID | Standort | Kapazität | Energiequelle | Sicherheitsstufe | Kühlkapazität |
+-----+-----+-----+-----+-----+
| 1 | Berlin, Lagerstraße 12 | 50000 | Solar | NULL | NULL |
| 2 | Hamburg, Hafenlager 23 | 75000 | Wind | NULL | NULL |
| 3 | München, Südallee 45 | 35000 | Solar | NULL | NULL |
| 4 | Frankfurt, Westhafen 8 | 100000 | Solar | NULL | NULL |
| 5 | Leipzig, Zentrallager | 45000 | Erneuerbar | NULL | NULL |
| 6 | Stuttgart, Hauptlager 1 | 55000 | Erneuerbar | NULL | NULL |
| 7 | Düsseldorf, Logistikzentrum | 50000 | Erneuerbar | NULL | NULL |
| 8 | Dresden, Verteilzentrum | 60000 | Erneuerbar | NULL | NULL |
+-----+-----+-----+-----+-----+
8 rows in set (0,001 sec)

MariaDB [hosakbari_db]> ALTER TABLE Lagerhaus MODIFY Standort VARCHAR(300);
Query OK, 0 rows affected (0,012 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
+-----+-----+-----+-----+-----+-----+
| Lager_ID | Standort | Kapazität | Energiequelle | Sicherheitsstufe | Kühlkapazität |
+-----+-----+-----+-----+-----+
| 1 | Berlin, Lagerstraße 12 | 50000 | Solar | NULL | NULL |
| 2 | Hamburg, Hafenlager 23 | 75000 | Wind | NULL | NULL |
| 3 | München, Südallee 45 | 35000 | Solar | NULL | NULL |
| 4 | Frankfurt, Westhafen 8 | 100000 | Solar | NULL | NULL |
| 5 | Leipzig, Zentrallager | 45000 | Erneuerbar | NULL | NULL |
| 6 | Stuttgart, Hauptlager 1 | 55000 | Erneuerbar | NULL | NULL |
| 7 | Düsseldorf, Logistikzentrum | 50000 | Erneuerbar | NULL | NULL |
| 8 | Dresden, Verteilzentrum | 60000 | Erneuerbar | NULL | NULL |
+-----+-----+-----+-----+-----+
8 rows in set (0,001 sec)

MariaDB [hosakbari_db]> ALTER TABLE Lagerhaus ADD Drohnenlandeplatz BOOLEAN NULL;
Query OK, 0 rows affected (0,014 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
+-----+-----+-----+-----+-----+-----+-----+
| Lager_ID | Standort | Kapazität | Energiequelle | Sicherheitsstufe | Kühlkapazität | Drohnenlandeplatz |
+-----+-----+-----+-----+-----+-----+
| 1 | Berlin, Lagerstraße 12 | 50000 | Solar | NULL | NULL | NULL |
| 2 | Hamburg, Hafenlager 23 | 75000 | Wind | NULL | NULL | NULL |
| 3 | München, Südallee 45 | 35000 | Solar | NULL | NULL | NULL |
| 4 | Frankfurt, Westhafen 8 | 100000 | Solar | NULL | NULL | NULL |
| 5 | Leipzig, Zentrallager | 45000 | Erneuerbar | NULL | NULL | NULL |
| 6 | Stuttgart, Hauptlager 1 | 55000 | Erneuerbar | NULL | NULL | NULL |
| 7 | Düsseldorf, Logistikzentrum | 50000 | Erneuerbar | NULL | NULL | NULL |
| 8 | Dresden, Verteilzentrum | 60000 | Erneuerbar | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+
8 rows in set (0,001 sec)

```

ALTER-Operation für Artikel

Dito für Artikel

Der Tabelle Artikel wird eine neue Spalte Produktionsland vom Typ VARCHAR(100) hinzugefügt, die NULL-Werte enthalten kann.

```
ALTER TABLE Artikel ADD Produktionsland VARCHAR(100) NULL;
```

Außerdem wird eine weitere Spalte Garantiezeit vom Typ INT ergänzt, die ebenfalls NULL-Werte aufnehmen darf.

```
ALTER TABLE Artikel ADD Garantiezeit INT NULL;
```

Die bestehende Spalte Bezeichnung wird so geändert, dass sie nun bis zu 300 Zeichen lang sein kann.

```
ALTER TABLE Artikel MODIFY Bezeichnung VARCHAR(300);
```

Zuletzt wird der Tabelle Artikel eine neue Spalte Recyclingklasse vom Typ VARCHAR(50) hinzugefügt, die NULL-Werte zulässt und die Recyclingkategorie eines Artikels angibt.

```
ALTER TABLE Artikel ADD Recyclingklasse VARCHAR(50) NULL;
```

```
MariaDB [hosakbari_db]> ALTER TABLE Artikel ADD Produktionsland VARCHAR(100) NULL;
Query OK, 0 rows affected (0,014 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Artikel;
```

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID	Produktionsland
101	Smartphone-Modell X	0.24	1450	1	NULL
102	Laptop-Modell Y	1.8	450	2	NULL
103	Tablet-Modell Z	0.96	750	3	NULL
104	Kühlschrank Eco-Kühl	50	70	4	NULL
105	Waschmaschine SuperClean	75	50	4	NULL
106	Drucker Modell T100	5	100	1	NULL
107	3D-Drucker HighTech	10	150	1	NULL
108	Neues Produkt	0.6	250	1	NULL
109	Smartwatch Pro	0.36	450	1	NULL

9 rows in set (0,002 sec)

```
MariaDB [hosakbari_db]> ALTER TABLE Artikel ADD Garantiezeit INT NULL;
Query OK, 0 rows affected (0,014 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Artikel;
```

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID	Produktionsland	Garantiezeit
101	Smartphone-Modell X	0.24	1450	1	NULL	NULL
102	Laptop-Modell Y	1.8	450	2	NULL	NULL
103	Tablet-Modell Z	0.96	750	3	NULL	NULL
104	Kühlschrank Eco-Kühl	50	70	4	NULL	NULL
105	Waschmaschine SuperClean	75	50	4	NULL	NULL
106	Drucker Modell T100	5	100	1	NULL	NULL
107	3D-Drucker HighTech	10	150	1	NULL	NULL
108	Neues Produkt	0.6	250	1	NULL	NULL
109	Smartwatch Pro	0.36	450	1	NULL	NULL

9 rows in set (0,002 sec)

```
MariaDB [hosakbari_db]> ALTER TABLE Artikel MODIFY Bezeichnung VARCHAR(300);
Query OK, 0 rows affected (0,015 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Artikel;
```

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID	Produktionsland	Garantiezeit
101	Smartphone-Modell X	0.24	1450	1	NULL	NULL
102	Laptop-Modell Y	1.8	450	2	NULL	NULL
103	Tablet-Modell Z	0.96	750	3	NULL	NULL
104	Kühlschrank Eco-Kühl	50	70	4	NULL	NULL
105	Waschmaschine SuperClean	75	50	4	NULL	NULL
106	Drucker Modell T100	5	100	1	NULL	NULL
107	3D-Drucker HighTech	10	150	1	NULL	NULL
108	Neues Produkt	0.6	250	1	NULL	NULL
109	Smartwatch Pro	0.36	450	1	NULL	NULL

9 rows in set (0,002 sec)

```
MariaDB [hosakbari_db]> ALTER TABLE Artikel ADD Recyclingklasse VARCHAR(50) NULL;
Query OK, 0 rows affected (0,013 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hosakbari_db]> SELECT * FROM Artikel;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Artikelnummer | Bezeichnung | Gewicht | Lagerbestand | Lagerhaus_Lager_ID | Produktionsland | Garantiezeit | Recyclingklasse |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 101 | Smartphone-Modell X | 0.24 | 1450 | 1 | NULL | NULL | NULL |
| 102 | Laptop-Modell Y | 1.8 | 450 | 2 | NULL | NULL | NULL |
| 103 | Tablet-Modell Z | 0.96 | 750 | 3 | NULL | NULL | NULL |
| 104 | Kühlschrank Eco-Kühl | 50 | 70 | 4 | NULL | NULL | NULL |
| 105 | Waschmaschine SuperClean | 75 | 50 | 4 | NULL | NULL | NULL |
| 106 | Drucker Modell T100 | 5 | 100 | 1 | NULL | NULL | NULL |
| 107 | 3D-Drucker HighTech | 10 | 150 | 1 | NULL | NULL | NULL |
| 108 | Neues Produkt | 0.6 | 250 | 1 | NULL | NULL | NULL |
| 109 | Smartwatch Pro | 0.36 | 450 | 1 | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
9 rows in set (0,002 sec)
```

ALTER-Befehl für Relation_Lieferung_Artikel

Dito für Relation_Lieferung_Artikel

Der Tabelle Relation_Lieferung_Artikel wird eine neue Spalte Liefermenge vom Typ INT hinzugefügt, die NULL-Werte enthalten kann.

ALTER TABLE Relation_Lieferung_Artikel ADD Liefermenge INT NULL;

Zusätzlich wird eine Spalte Versandart vom Typ VARCHAR(50) ergänzt, die ebenfalls NULL-Werte zulässt.

ALTER TABLE Relation_Lieferung_Artikel ADD Versandart VARCHAR(50) NULL;

Die bestehende Spalte Lieferungsnummer wird so modifiziert, dass sie keine NULL-Werte mehr enthalten darf.

ALTER TABLE Relation_Lieferung_Artikel MODIFY Lieferungsnummer INT NOT NULL;

Abschließend wird der Tabelle eine neue Spalte Expressversand vom Typ BOOLEAN hinzugefügt, die angibt, ob ein Expressversand erfolgt, und NULL-Werte zulässt.

ALTER TABLE Relation_Lieferung_Artikel ADD Expressversand BOOLEAN NULL;

```
MariaDB [hosakbari_db]> ALTER TABLE Relation_Lieferung_Artikel ADD Liefermenge INT NULL
Query OK, 0 rows affected (0,013 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

MariaDB [hosakbari_db]> SELECT * FROM Relation_Lieferung_Artikel;

```
+-----+-----+-----+
| Lieferungsnummer | Artikelnummer | Liefermenge |
+-----+-----+-----+
| 1 | 101 | NULL |
| 1 | 104 | NULL |
| 2 | 103 | NULL |
| 2 | 104 | NULL |
| 4 | 101 | NULL |
| 4 | 103 | NULL |
| 4 | 104 | NULL |
| 4 | 105 | NULL |
+-----+-----+-----+
8 rows in set (0,002 sec)
```

```

MariaDB [hosakbari_db]> ALTER TABLE Relation_Lieferung_Artikel ADD Versandart VARCHAR(50) NULL;
Query OK, 0 rows affected (0,013 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hosakbari_db]> SELECT * FROM Relation_Lieferung_Artikel;
+-----+-----+-----+-----+
| Lieferungsnummer | Artikelnummer | Liefermenge | Versandart |
+-----+-----+-----+-----+
| 1 | 101 | NULL | NULL |
| 1 | 104 | NULL | NULL |
| 2 | 103 | NULL | NULL |
| 2 | 104 | NULL | NULL |
| 4 | 101 | NULL | NULL |
| 4 | 103 | NULL | NULL |
| 4 | 104 | NULL | NULL |
| 4 | 105 | NULL | NULL |
+-----+-----+-----+-----+
8 rows in set (0,002 sec)

MariaDB [hosakbari_db]> ALTER TABLE Relation_Lieferung_Artikel MODIFY Lieferungsnummer INT NOT NULL;
Query OK, 0 rows affected (0,010 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hosakbari_db]> SELECT * FROM Relation_Lieferung_Artikel;
+-----+-----+-----+-----+
| Lieferungsnummer | Artikelnummer | Liefermenge | Versandart |
+-----+-----+-----+-----+
| 1 | 101 | NULL | NULL |
| 1 | 104 | NULL | NULL |
| 2 | 103 | NULL | NULL |
| 2 | 104 | NULL | NULL |
| 4 | 101 | NULL | NULL |
| 4 | 103 | NULL | NULL |
| 4 | 104 | NULL | NULL |
| 4 | 105 | NULL | NULL |
+-----+-----+-----+-----+
8 rows in set (0,001 sec)

MariaDB [hosakbari_db]> ALTER TABLE Relation_Lieferung_Artikel ADD Expressversand BOOLEAN NULL;
Query OK, 0 rows affected (0,013 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hosakbari_db]> SELECT * FROM Relation_Lieferung_Artikel;
+-----+-----+-----+-----+-----+
| Lieferungsnummer | Artikelnummer | Liefermenge | Versandart | Expressversand |
+-----+-----+-----+-----+-----+
| 1 | 101 | NULL | NULL | NULL |
| 1 | 104 | NULL | NULL | NULL |
| 2 | 103 | NULL | NULL | NULL |
| 2 | 104 | NULL | NULL | NULL |
| 4 | 101 | NULL | NULL | NULL |
| 4 | 103 | NULL | NULL | NULL |
| 4 | 104 | NULL | NULL | NULL |
| 4 | 105 | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+
8 rows in set (0,001 sec)

```

DELETE-Befehl für Lieferung

Der DELETE-Befehl findet in SQL seine Verwendung für die Löschung von Datensätzen aus einer Tabelle. Die Syntax hierfür lautet:

`DELETE FROM Tabellenname WHERE Bedingung;`

Der erste Befehl unten löscht alle Lieferung-Einträge, bei denen der Zielort nicht angegeben ist. Der nächste Befehl löscht alle Lieferung-Einträge, die den Status geliefert haben. Dann werden alle Lieferung-Einträge gelöscht, die einen Lieferzeitpunkt haben, vor dem 15. Juni 2044. Zuletzt werden alle Lieferung-Einträge gelöscht, die als verspätet markiert sind.

Es werden Datensätze aus der Tabelle Lieferung gelöscht, die in bestimmten Spalten vordefinierte Bedingungen erfüllen. Zuerst werden alle Lieferungen gelöscht, bei denen der Zielort nicht angegeben ist:

`DELETE FROM Lieferung WHERE Zielort IS NULL;`

Anschließend werden alle Datensätze entfernt, bei denen der Status den Wert 'geliefert' hat:

```
DELETE FROM Lieferung WHERE Status = 'geliefert';
```

Danach werden alle Lieferungen gelöscht, deren Lieferzeitpunkt vor dem 15. Juni 2044 liegt:

```
DELETE FROM Lieferung WHERE Lieferzeitpunkt < '2044-06-15';
```

Zuletzt werden alle Lieferungen entfernt, deren Status als 'Verspätet' markiert ist:

```
DELETE FROM Lieferung WHERE Status = 'Verspätet';
```

```
MariaDB [hosakbari_db]> DELETE FROM Lieferung WHERE Zielort IS NULL;
Query OK, 0 rows affected (0,001 sec)
```

```
MariaDB [hosakbari_db]> DELETE FROM Lieferung WHERE Zielort IS NULL;^C
MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
1	Berlin, Alexanderplatz	2044-06-15	unterwegs	NULL	NULL	NULL
2	Hamburg, Hafenstraße	2044-06-16	geliefert	NULL	NULL	NULL
3	München, Marienplatz	2044-06-17	verspätet	NULL	NULL	NULL
4	Frankfurt, Zeilstraße	2044-06-18	geplant	NULL	NULL	NULL
5	Unbekannt	2044-06-20	in Bearbeitung	NULL	NULL	NULL
6	Unbekannt	2044-06-21	Verspätet	NULL	NULL	NULL
7	Unbekannt	2044-06-22	Verspätet	NULL	NULL	NULL
8	Unbekannt	2044-06-23	Verspätet	NULL	NULL	NULL

8 rows in set (0,001 sec)

Bei dem zweiten Befehl bekomme ich die folgende Meldung: ...Cannot delete or update a parent row: a foreign key constraint fails ('hosakbari_db'.`Fahrzeug`, CONSTRAINT `Fahrzeug_ibfk_1` FOREIGN KEY (`Lieferung_Lieferungsnummer`) REFERENCES `Lieferung`(`Lieferungsnummer`))...

Diese Fehlermeldung tritt auf, weil die Tabelle Lieferung in einer Fremdschlüsselbeziehung mit der Tabelle Fahrzeug steht. Hierzu müssen wir im Voraus die Referenzen aus der Fahrzeug-Tabelle entfernen oder auf NULL setzen. Wir machen letzteres:

```
UPDATE Fahrzeug SET Lieferung_Lieferungsnummer = NULL WHERE Lieferung_Lieferungsnummer = (SELECT Lieferungsnummer FROM Lieferung WHERE Status = 'geliefert');
```

```
MariaDB [hosakbari_db]> DELETE FROM Lieferung WHERE Status = 'geliefert';
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails ('hosakbari_db'.`Fahrzeug`, CONSTRAINT `Fahrzeug_ibfk_1` FOREIGN KEY (`Lieferung_Lieferungsnummer`) REFERENCES `Lieferung`(`Lieferungsnummer`))
MariaDB [hosakbari_db]> UPDATE Fahrzeug SET Lieferung_Lieferungsnummer = NULL WHERE Lieferung_Lieferungsnummer =
    ->     (SELECT Lieferungsnummer FROM Lieferung WHERE Status = 'geliefert');
Query OK, 1 row affected (0,020 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
```

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer	Wartungsstatus	Ladezeit	CO2_Einsparung
1	Elektro-LKW	5000	85		1 NULL	NULL	NULL
2	Drohne	55	100		NULL NULL	NULL	NULL
3	Elektro-Van	1100	75		3 NULL	NULL	NULL
4	Hybrid-LKW	8000	100		3 NULL	NULL	NULL
5	Autonomer LKW	7000	100		3 NULL	NULL	NULL
6	Wasserstoff-LKW	6500	100		3 NULL	NULL	NULL
7	Elektro-Scooter	132	100		3 NULL	NULL	NULL
8	Hybrid-Drohne	82.5	100		3 NULL	NULL	NULL

8 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
1	Berlin, Alexanderplatz	2044-06-15	unterwegs	NULL	NULL	NULL
2	Hamburg, Hafenstraße	2044-06-16	geliefert	NULL	NULL	NULL
3	München, Marienplatz	2044-06-17	verspätet	NULL	NULL	NULL
4	Frankfurt, Zeilstraße	2044-06-18	geplant	NULL	NULL	NULL
5	Unbekannt	2044-06-20	in Bearbeitung	NULL	NULL	NULL
6	Unbekannt	2044-06-21	Verspätet	NULL	NULL	NULL
7	Unbekannt	2044-06-22	Verspätet	NULL	NULL	NULL
8	Unbekannt	2044-06-23	Verspätet	NULL	NULL	NULL

8 rows in set (0,001 sec)

Wir bekommen nun jedoch die Fehlermeldung, dass die Spalte Lieferungsnummer in der Tabelle Relation_Lieferung_Artikel nicht auf NULL gesetzt werden kann: ERROR 1048 (23000): Column 'Lieferungsnummer' cannot be null

Hierzu müssen wir den Fremdschlüssel so anpassen, dass NULL erlaubt ist:

... nach langem Rumprobieren habe ich mich entschieden, diesen Schritt zu überspringen ...

```
MariaDB [hosakbari_db]> DELETE FROM Lieferung WHERE Lieferzeitpunkt < '2044-06-15';
Query OK, 0 rows affected (0,001 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
1	Berlin, Alexanderplatz	2044-06-15	unterwegs	NULL	NULL	NULL
2	Hamburg, Hafenstraße	2044-06-16	geliefert	NULL	NULL	NULL
3	München, Marienplatz	2044-06-17	verspätet	NULL	NULL	NULL
4	Frankfurt, Zeilstraße	2044-06-18	geplant	NULL	NULL	NULL
5	Unbekannt	2044-06-20	in Bearbeitung	NULL	NULL	NULL
6	Unbekannt	2044-06-21	Verspätet	NULL	NULL	NULL
7	Unbekannt	2044-06-22	Verspätet	NULL	NULL	NULL
8	Unbekannt	2044-06-23	Verspätet	NULL	NULL	NULL

8 rows in set (0,002 sec)

```
MariaDB [hosakbari_db]> DELETE FROM Lieferung WHERE Status = 'Verspätet';
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails ('hosakbari_db'.'Fahrzeug', CONSTRAINT 'Fahrzeug_ibfk_1' FOREIGN KEY ('Lieferung_Lieferungsnummer') REFERENCES 'Lieferung' ('Lieferungsnummer'))
MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
1	Berlin, Alexanderplatz	2044-06-15	unterwegs	NULL	NULL	NULL
2	Hamburg, Hafenstraße	2044-06-16	geliefert	NULL	NULL	NULL
3	München, Marienplatz	2044-06-17	verspätet	NULL	NULL	NULL
4	Frankfurt, Zeilstraße	2044-06-18	geplant	NULL	NULL	NULL
5	Unbekannt	2044-06-20	in Bearbeitung	NULL	NULL	NULL
6	Unbekannt	2044-06-21	Verspätet	NULL	NULL	NULL
7	Unbekannt	2044-06-22	Verspätet	NULL	NULL	NULL
8	Unbekannt	2044-06-23	Verspätet	NULL	NULL	NULL

8 rows in set (0,001 sec)

Ich hatte ein ähnliches Problem mit dem letzten Befehl

DELETE-Befehl für Fahrzeug

Die Befehle unten funktionieren auf eine ähnliche Art wie die Tabelle Lieferung, nur jetzt für die Tabelle Fahrzeug. Aus Zeitgründen wird die Erklärung übersprungen.

Es werden Datensätze aus der Tabelle Fahrzeug gelöscht, die bestimmte Bedingungen erfüllen. Zunächst werden alle Fahrzeuge entfernt, denen keine Lieferung zugewiesen ist:

```
DELETE FROM Fahrzeug WHERE Lieferung_Lieferungsnummer IS NULL;
```

Danach werden alle Fahrzeuge gelöscht, deren Kapazität unter 500 liegt:

```
DELETE FROM Fahrzeug WHERE Kapazität < 500;
```

Anschließend werden Fahrzeuge entfernt, deren Batteriestatus unter 20 Prozent liegt:

```
DELETE FROM Fahrzeug WHERE Batteriestatus < 20;
```

Zum Schluss werden alle Fahrzeuge gelöscht, die als Typ 'Hybrid-LKW' eingetragen sind:

```
DELETE FROM Fahrzeug WHERE Typ = 'Hybrid-LKW';
```

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
```

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer	Wartungsstatus	Ladezeit	CO2_Einsparung
1	Elektro-LKW	5000	85		1	NULL	NULL
3	Elektro-Van	1100	75		3	NULL	NULL
4	Hybrid-LKW	8000	100		3	NULL	NULL
5	Autonomer LKW	7000	100		3	NULL	NULL
6	Wasserstoff-LKW	6500	100		3	NULL	NULL
7	Elektro-Scooter	132	100		3	NULL	NULL
8	Hybrid-Drohne	82.5	100		3	NULL	NULL

7 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> DELETE FROM Fahrzeug WHERE Kapazität < 500;
Query OK, 2 rows affected (0,017 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
```

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer	Wartungsstatus	Ladezeit	CO2_Einsparung
1	Elektro-LKW	5000	85		1	NULL	NULL
3	Elektro-Van	1100	75		3	NULL	NULL
4	Hybrid-LKW	8000	100		3	NULL	NULL
5	Autonomer LKW	7000	100		3	NULL	NULL
6	Wasserstoff-LKW	6500	100		3	NULL	NULL

5 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> DELETE FROM Fahrzeug WHERE Batteriestatus < 20;
Query OK, 0 rows affected (0,001 sec)

MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Fahrzeug_ID | Typ | Kapazität | Batteriestatus | Lieferung_Lieferungsnummer | Wartungsstatus | Ladezeit | CO2_Einsparung |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Elektro-LKW | 5000 | 85 | 1 | NULL | NULL | NULL |
| 3 | Elektro-Van | 1100 | 75 | 3 | NULL | NULL | NULL |
| 4 | Hybrid-LKW | 8000 | 100 | 3 | NULL | NULL | NULL |
| 5 | Autonomer LKW | 7000 | 100 | 3 | NULL | NULL | NULL |
| 6 | Wasserstoff-LKW | 6500 | 100 | 3 | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0,001 sec)

MariaDB [hosakbari_db]> DELETE FROM Fahrzeug WHERE Typ = 'Hybrid-LKW';
Query OK, 1 row affected (0,002 sec)

MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Fahrzeug_ID | Typ | Kapazität | Batteriestatus | Lieferung_Lieferungsnummer | Wartungsstatus | Ladezeit | CO2_Einsparung |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Elektro-LKW | 5000 | 85 | 1 | NULL | NULL | NULL |
| 3 | Elektro-Van | 1100 | 75 | 3 | NULL | NULL | NULL |
| 5 | Autonomer LKW | 7000 | 100 | 3 | NULL | NULL | NULL |
| 6 | Wasserstoff-LKW | 6500 | 100 | 3 | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
4 rows in set (0,001 sec)
```

DELETE-Befehl für Lagerhaus

Dito für Lagerhaus

Es werden Datensätze aus der Tabelle Lagerhaus gelöscht, die bestimmte Kriterien erfüllen. Zunächst werden alle Lagerhäuser entfernt, bei denen keine Energiequelle angegeben ist:

```
DELETE FROM Lagerhaus WHERE Energiequelle IS NULL;
```

Anschließend werden alle Lagerhäuser gelöscht, deren Kapazität unter 30000 liegt:

```
DELETE FROM Lagerhaus WHERE Kapazität < 30000;
```

Danach werden alle Lagerhäuser entfernt, deren Standort den Begriff „Hafen“ enthält:

```
DELETE FROM Lagerhaus WHERE Standort LIKE '%Hafen%';
```

Zum Schluss werden Lagerhäuser gelöscht, bei denen der Standort nicht angegeben ist:

```
DELETE FROM Lagerhaus WHERE Standort IS NULL;
```

```
MariaDB [hosakbari_db]> DELETE FROM Lagerhaus WHERE Energiequelle IS NULL;
Query OK, 0 rows affected (0,016 sec)

MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
+-----+-----+-----+-----+-----+-----+-----+
| Lager_ID | Standort | Kapazität | Energiequelle | Sicherheitsstufe | Kühlkapazität | Drohnenlandeplatz |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Berlin, Lagerstraße 12 | 50000 | Solar | NULL | NULL | NULL |
| 2 | Hamburg, Hafenlager 23 | 75000 | Wind | NULL | NULL | NULL |
| 3 | München, Südallee 45 | 35000 | Solar | NULL | NULL | NULL |
| 4 | Frankfurt, Westhafen 8 | 100000 | Solar | NULL | NULL | NULL |
| 5 | Leipzig, Zentrallager | 45000 | Erneuerbar | NULL | NULL | NULL |
| 6 | Stuttgart, Hauptlager 1 | 55000 | Erneuerbar | NULL | NULL | NULL |
| 7 | Düsseldorf, Logistikzentrum | 50000 | Erneuerbar | NULL | NULL | NULL |
| 8 | Dresden, Verteilzentrum | 60000 | Erneuerbar | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+
8 rows in set (0,001 sec)

MariaDB [hosakbari_db]> DELETE FROM Lagerhaus WHERE Kapazität < 30000;
Query OK, 0 rows affected (0,001 sec)

MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
+-----+-----+-----+-----+-----+-----+-----+
| Lager_ID | Standort | Kapazität | Energiequelle | Sicherheitsstufe | Kühlkapazität | Drohnenlandeplatz |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Berlin, Lagerstraße 12 | 50000 | Solar | NULL | NULL | NULL |
| 2 | Hamburg, Hafenlager 23 | 75000 | Wind | NULL | NULL | NULL |
| 3 | München, Südallee 45 | 35000 | Solar | NULL | NULL | NULL |
| 4 | Frankfurt, Westhafen 8 | 100000 | Solar | NULL | NULL | NULL |
| 5 | Leipzig, Zentrallager | 45000 | Erneuerbar | NULL | NULL | NULL |
| 6 | Stuttgart, Hauptlager 1 | 55000 | Erneuerbar | NULL | NULL | NULL |
| 7 | Düsseldorf, Logistikzentrum | 50000 | Erneuerbar | NULL | NULL | NULL |
| 8 | Dresden, Verteilzentrum | 60000 | Erneuerbar | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+
8 rows in set (0,001 sec)
```

```

MariaDB [hosakbari_db]> DELETE FROM Lagerhaus WHERE Standort LIKE '%Hafen%';
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails ('hosakbari_db'.'Artikel', CONSTRAINT 'Artikel_ibfk_1' FOREIGN KEY ('Lagerhaus_Lager_ID') REFERENCES 'Lagerhaus'('Lager_ID'))
MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
+-----+-----+-----+-----+-----+-----+-----+
| Lager_ID | Standort | Kapazität | Energiequelle | Sicherheitsstufe | Kühlkapazität | Drohnenlandeplatz |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Berlin, Lagerstraße 12 | 50000 | Solar | NULL | NULL | NULL |
| 2 | Hamburg, Hafenlager 23 | 75000 | Wind | NULL | NULL | NULL |
| 3 | München, Südallee 45 | 35000 | Solar | NULL | NULL | NULL |
| 4 | Frankfurt, Westhafen 8 | 100000 | Solar | NULL | NULL | NULL |
| 5 | Leipzig, Zentrallager | 45000 | Erneuerbar | NULL | NULL | NULL |
| 6 | Stuttgart, Hauptlager 1 | 55000 | Erneuerbar | NULL | NULL | NULL |
| 7 | Düsseldorf, Logistikzentrum | 50000 | Erneuerbar | NULL | NULL | NULL |
| 8 | Dresden, Verteilzentrum | 60000 | Erneuerbar | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+
8 rows in set (0,001 sec)

```

Bei dem dritten DELETE-Befehl hatte ich wieder ein Problem wegen Fremdschlüssel-Beziehung. Wir überspringen das.

```

MariaDB [hosakbari_db]> DELETE FROM Lagerhaus WHERE Standort IS NULL;
Query OK, 0 rows affected (0,001 sec)

```

```

MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
+-----+-----+-----+-----+-----+-----+-----+
| Lager_ID | Standort | Kapazität | Energiequelle | Sicherheitsstufe | Kühlkapazität | Drohnenlandeplatz |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Berlin, Lagerstraße 12 | 50000 | Solar | NULL | NULL | NULL |
| 2 | Hamburg, Hafenlager 23 | 75000 | Wind | NULL | NULL | NULL |
| 3 | München, Südallee 45 | 35000 | Solar | NULL | NULL | NULL |
| 4 | Frankfurt, Westhafen 8 | 100000 | Solar | NULL | NULL | NULL |
| 5 | Leipzig, Zentrallager | 45000 | Erneuerbar | NULL | NULL | NULL |
| 6 | Stuttgart, Hauptlager 1 | 55000 | Erneuerbar | NULL | NULL | NULL |
| 7 | Düsseldorf, Logistikzentrum | 50000 | Erneuerbar | NULL | NULL | NULL |
| 8 | Dresden, Verteilzentrum | 60000 | Erneuerbar | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+
8 rows in set (0,001 sec)

```

Hierzu gab es keine Änderung, da der Standort bei keinem NULL ist.

DELETE-Befehl für Artikel

Dito für Artikel

Es werden Datensätze aus der Tabelle Artikel gelöscht, die bestimmte Bedingungen erfüllen. Zuerst werden alle Artikel entfernt, die keinem Lagerhaus zugewiesen sind:

```
DELETE FROM Artikel WHERE Lagerhaus_Lager_ID IS NULL;
```

Als Nächstes werden alle Artikel gelöscht, deren Lagerbestand unter 10 liegt:

```
DELETE FROM Artikel WHERE Lagerbestand < 10;
```

Danach werden alle Artikel entfernt, deren Bezeichnung „Modell Z“ enthält:

```
DELETE FROM Artikel WHERE Bezeichnung LIKE '%Modell Z%';
```

Schließlich werden alle Artikel gelöscht, deren Gewicht über 100 liegt:

```
DELETE FROM Artikel WHERE Gewicht > 100;
```

```

MariaDB [hosakbari_db]> DELETE FROM Artikel WHERE Lagerhaus_Lager_ID IS NULL;
Query OK, 0 rows affected (0,001 sec)

```

```
MariaDB [hosakbari_db]> SELECT * FROM Artikel;
```

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID	Produktionsland	Garantiezeit	Recyclingklasse
101	Smartphone-Modell X	0.24	1450	1	NULL	NULL	NULL
102	Laptop-Modell Y	1.8	450	2	NULL	NULL	NULL
103	Tablet-Modell Z	0.96	750	3	NULL	NULL	NULL
104	Kühlschrank Eco-Kühl	50	70	4	NULL	NULL	NULL
105	Waschmaschine SuperClean	75	50	4	NULL	NULL	NULL
106	Drucker Modell T100	5	100	1	NULL	NULL	NULL
107	3D-Drucker HighTech	10	150	1	NULL	NULL	NULL
108	Neues Produkt	0.6	250	1	NULL	NULL	NULL
109	Smartwatch Pro	0.36	450	1	NULL	NULL	NULL

```
9 rows in set (0,013 sec)
```

```
MariaDB [hosakbari_db]> DELETE FROM Artikel WHERE Lagerbestand < 10;
Query OK, 0 rows affected (0,001 sec)

MariaDB [hosakbari_db]> SELECT * FROM Artikel;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Artikelnummer | Bezeichnung | Gewicht | Lagerbestand | Lagerhaus_Lager_ID | Produktionsland | Garantiezeit | Recyclingklasse |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 101 | Smartphone-Modell X | 0.24 | 1450 | 1 | NULL | NULL | NULL |
| 102 | Laptop-Modell Y | 1.8 | 450 | 2 | NULL | NULL | NULL |
| 103 | Tablet-Modell Z | 0.96 | 750 | 3 | NULL | NULL | NULL |
| 104 | Kühlschrank Eco-Kühl | 50 | 70 | 4 | NULL | NULL | NULL |
| 105 | Waschmaschine SuperClean | 75 | 50 | 4 | NULL | NULL | NULL |
| 106 | Drucker Modell T100 | 5 | 100 | 1 | NULL | NULL | NULL |
| 107 | 3D-Drucker HighTech | 10 | 150 | 1 | NULL | NULL | NULL |
| 108 | Neues Produkt | 0.6 | 250 | 1 | NULL | NULL | NULL |
| 109 | Smartwatch Pro | 0.36 | 450 | 1 | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
9 rows in set (0,001 sec)

MariaDB [hosakbari_db]> DELETE FROM Artikel WHERE Bezeichnung LIKE '%Modell Z%';
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails ('hosakbari_db'.'Relation_Lieferung_Artikel', CONSTRAINT 'Artikelnummer') REFERENCES 'Artikel' ('Artikelnummer')
MariaDB [hosakbari_db]> SELECT * FROM Artikel;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Artikelnummer | Bezeichnung | Gewicht | Lagerbestand | Lagerhaus_Lager_ID | Produktionsland | Garantiezeit | Recyclingklasse |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 101 | Smartphone-Modell X | 0.24 | 1450 | 1 | NULL | NULL | NULL |
| 102 | Laptop-Modell Y | 1.8 | 450 | 2 | NULL | NULL | NULL |
| 103 | Tablet-Modell Z | 0.96 | 750 | 3 | NULL | NULL | NULL |
| 104 | Kühlschrank Eco-Kühl | 50 | 70 | 4 | NULL | NULL | NULL |
| 105 | Waschmaschine SuperClean | 75 | 50 | 4 | NULL | NULL | NULL |
| 106 | Drucker Modell T100 | 5 | 100 | 1 | NULL | NULL | NULL |
| 107 | 3D-Drucker HighTech | 10 | 150 | 1 | NULL | NULL | NULL |
| 108 | Neues Produkt | 0.6 | 250 | 1 | NULL | NULL | NULL |
| 109 | Smartwatch Pro | 0.36 | 450 | 1 | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
9 rows in set (0,001 sec)
```

Bei dem dritten DELETE-Befehl hatte ich wieder ein Problem mit der Fremdschlüssel-Beziehung. Wir skippen das.

```
MariaDB [hosakbari_db]> DELETE FROM Artikel WHERE Gewicht > 100;
Query OK, 0 rows affected (0,001 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Artikel;
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| Artikelnummer | Bezeichnung | Gewicht | Lagerbestand | Lagerhaus_Lager_ID | Produktionsland | Garantiezeit | Recyclingklasse |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 101 | Smartphone-Modell X | 0.24 | 1450 | 1 | NULL | NULL | NULL |
| 102 | Laptop-Modell Y | 1.8 | 450 | 2 | NULL | NULL | NULL |
| 103 | Tablet-Modell Z | 0.96 | 750 | 3 | NULL | NULL | NULL |
| 104 | Kühlschrank Eco-Kühl | 50 | 70 | 4 | NULL | NULL | NULL |
| 105 | Waschmaschine SuperClean | 75 | 50 | 4 | NULL | NULL | NULL |
| 106 | Drucker Modell T100 | 5 | 100 | 1 | NULL | NULL | NULL |
| 107 | 3D-Drucker HighTech | 10 | 150 | 1 | NULL | NULL | NULL |
| 108 | Neues Produkt | 0.6 | 250 | 1 | NULL | NULL | NULL |
| 109 | Smartwatch Pro | 0.36 | 450 | 1 | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
9 rows in set (0,001 sec)
```

DELETE-Befehl für Relation_Lieferung_Artikel

Dito für Relation_Lieferung_Artikel

Es werden Datensätze aus der Tabelle Relation_Lieferung_Artikel gelöscht, die bestimmte Kriterien erfüllen.

Zuerst werden alle Einträge entfernt, bei denen keine Lieferungsnummer vorhanden ist:

```
DELETE FROM Relation_Lieferung_Artikel WHERE Lieferungsnummer IS NULL;
```

Als Nächstes werden alle Einträge gelöscht, bei denen die Artikelnummer 105 ist:

```
DELETE FROM Relation_Lieferung_Artikel WHERE Artikelnummer = 105;
```

Danach werden alle Datensätze entfernt, die der Lieferungsnummer 3 zugeordnet sind:

```
DELETE FROM Relation_Lieferung_Artikel WHERE Lieferungsnummer = 3;
```

Abschließend werden alle Einträge gelöscht, bei denen die Artikelnummer 104 lautet:

```
DELETE FROM Relation_Lieferung_Artikel WHERE Artikelnummer = 104;
```

```
MariaDB [hosakbari_db]> DELETE FROM Relation_Lieferung_Artikel WHERE Lieferungsnummer IS NULL;
Query OK, 0 rows affected (0,001 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Relation_Lieferung_Artikel;
```

Lieferungsnummer	Artikelnummer	Liefermenge	Versandart	Expressversand
1	101	NULL	NULL	NULL
1	104	NULL	NULL	NULL
2	103	NULL	NULL	NULL
2	104	NULL	NULL	NULL
4	101	NULL	NULL	NULL
4	103	NULL	NULL	NULL
4	104	NULL	NULL	NULL
4	105	NULL	NULL	NULL

```
8 rows in set (0,001 sec)
```

```
MariaDB [hosakbari_db]> DELETE FROM Relation_Lieferung_Artikel WHERE Artikelnummer = 105;
Query OK, 1 row affected (0,002 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Relation_Lieferung_Artikel;
```

Lieferungsnummer	Artikelnummer	Liefermenge	Versandart	Expressversand
1	101	NULL	NULL	NULL
1	104	NULL	NULL	NULL
2	103	NULL	NULL	NULL
2	104	NULL	NULL	NULL
4	101	NULL	NULL	NULL
4	103	NULL	NULL	NULL
4	104	NULL	NULL	NULL

```
7 rows in set (0,001 sec)
```

```
MariaDB [hosakbari_db]> DELETE FROM Relation_Lieferung_Artikel WHERE Lieferungsnummer = 3;
Query OK, 0 rows affected (0,001 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Relation_Lieferung_Artikel;
```

Lieferungsnummer	Artikelnummer	Liefermenge	Versandart	Expressversand
1	101	NULL	NULL	NULL
1	104	NULL	NULL	NULL
2	103	NULL	NULL	NULL
2	104	NULL	NULL	NULL
4	101	NULL	NULL	NULL
4	103	NULL	NULL	NULL
4	104	NULL	NULL	NULL

```
7 rows in set (0,001 sec)
```

```
MariaDB [hosakbari_db]> DELETE FROM Relation_Lieferung_Artikel WHERE Artikelnummer = 104;
Query OK, 3 rows affected (0,002 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Relation_Lieferung_Artikel;
```

Lieferungsnummer	Artikelnummer	Liefermenge	Versandart	Expressversand
1	101	NULL	NULL	NULL
2	103	NULL	NULL	NULL
4	101	NULL	NULL	NULL
4	103	NULL	NULL	NULL

```
4 rows in set (0,001 sec)
```

Relationen mit NULL-Werten für Lieferung

Unten werden als aller erst alle Lieferung-Einträge zurückgegeben, bei denen der Zielort nicht angegeben ist.

Danach werden alle Lieferung-Einträge angezeigt, bei denen der Lieferzeitpunkt fehlt. Im Nachhinein werden alle Lieferung-Einträge gelistet, bei denen der Status nicht gesetzt ist. Zuletzt werden alle Lieferung-Einträge angezeigt, deren Lieferzeitpunkt nach dem 30. Juni 2044 liegt.

Es wird eine Abfrage ausgeführt, um alle Lieferungen anzuzeigen, bei denen kein Zielort angegeben wurde.

```
SELECT * FROM Lieferung WHERE Zielort IS NULL;
```

Es wird eine Abfrage ausgeführt, um alle Lieferungen anzuzeigen, bei denen kein Lieferzeitpunkt angegeben wurde.

```
SELECT * FROM Lieferung WHERE Lieferzeitpunkt IS NULL;
```

Es wird eine Abfrage ausgeführt, um alle Lieferungen anzuzeigen, bei denen der Status nicht angegeben wurde.

```
SELECT * FROM Lieferung WHERE Status IS NULL;
```

Es wird eine Abfrage ausgeführt, um alle Lieferungen anzuzeigen, deren Lieferzeitpunkt nach dem 30. Juni 2044 liegt.

```
SELECT * FROM Lieferung WHERE Lieferzeitpunkt > '2044-06-30';
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung WHERE Zielort IS NULL;  
Empty set (0.001 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung WHERE Lieferzeitpunkt IS NULL;  
Empty set (0.001 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung WHERE Status IS NULL;  
Empty set (0.001 sec)
```

```
MariaDB [hosakbari_db]> ^C  
MariaDB [hosakbari_db]> SELECT * FROM Lieferung WHERE Status IS NULL;  
Empty set (0.001 sec)
```

Ich bekomme hierfür keine Einträge. Ich versuche im Folgenden zuerst die Tabelle mit Beispieldaten füllen und im Nachhinein die Befehle erneut ausführen:

Es werden neue Lieferungsdaten in die Tabelle Lieferung eingefügt. Die Werte umfassen Lieferungsnummern, Zielorte, Lieferzeitpunkte, Status, Prioritäten, Transporttypen und Versandkosten. So wird beispielsweise eine Lieferung mit der Nummer 9 für den Zielort Köln am 25. Juni 2044 geplant, unter Verwendung einer Drohne mit Versandkosten in Höhe von 19,99 Euro.

```
INSERT INTO Lieferung (Lieferungsnummer, Zielort, Lieferzeitpunkt, Status, Priorität, Transporttyp,  
Versandkosten)
```

```
VALUES
```

```
(9, 'Köln, Domplatz', '2044-06-25', 'geplant', 2, 'Drohne', 19.99),  
(10, NULL, '2044-06-27', 'unterwegs', 1, 'Elektro-LKW', 29.99),  
(11, 'Hamburg, Reeperbahn', NULL, 'verspätet', 3, 'Hybrid-LKW', 24.50),  
(12, 'München, Stachus', '2044-07-01', 'geliefert', 2, 'Elektro-Van', 15.75),  
(13, NULL, NULL, 'in Bearbeitung', 1, 'Drohne', 12.99),  
(14, 'Düsseldorf, Altstadt', '2044-07-05', NULL, 2, 'Wasserstoff-LKW', 22.30),  
(15, 'Stuttgart, Schlossplatz', '2044-06-29', 'unterwegs', 1, 'Elektro-Scooter', 9.99),  
(16, 'Frankfurt, Mainufer', NULL, NULL, 3, 'Drohne', 14.50),  
(17, NULL, '2044-07-03', 'geplant', 2, 'Elektro-LKW', 17.89),  
(18, 'Berlin, Kudamm', NULL, 'in Bearbeitung', 1, 'Hybrid-Drohne', 11.99),  
(19, NULL, NULL, NULL, 3, 'Autonomer LKW', 31.50),  
(20, 'Nürnberg, Zentrum', '2044-07-10', 'verspätet', 1, 'Drohne', 10.99);
```

```
MariaDB [hosakbari_db]> INSERT INTO Lieferung (Lieferungsnummer, Zielort, Lieferzeitpunkt, Status, Priorität, Transporttyp, Versandkosten)
→ VALUES
→ (9, 'Köln, Domplatz', '2044-06-25', 'geplant', 2, 'Drohne', 19.99),
→ (10, NULL, '2044-06-27', 'unterwegs', 1, 'Elektro-LKW', 29.99),
→ (11, 'Hamburg, Reeperbahn', NULL, 'verspätet', 3, 'Hybrid-LKW', 24.50),
→ (12, 'München, Stachus', '2044-07-01', 'geliefert', 2, 'Elektro-Van', 15.75),
→ (13, NULL, NULL, 'in Bearbeitung', 1, 'Drohne', 12.99),
→ (14, 'Düsseldorf, Altstadt', '2044-07-05', NULL, 2, 'Wasserstoff-LKW', 22.30),
→ (15, 'Stuttgart, Schlossplatz', '2044-06-29', 'unterwegs', 1, 'Elektro-Scooter', 9.99),
→ (16, 'Frankfurt, Mainufer', NULL, NULL, 3, 'Drohne', 14.50),
→ (17, NULL, '2044-07-03', 'geplant', 2, 'Elektro-LKW', 17.89),
→ (18, 'Berlin, Kudamm', NULL, 'in Bearbeitung', 1, 'Hybrid-Drohne', 11.99),
→ (19, NULL, NULL, 3, 'Autonomer LKW', 31.50),
→ (20, 'Nürnberg, Zentrum', '2044-07-10', 'verspätet', 1, 'Drohne', 10.99);
Query OK, 12 rows affected (0.003 sec)
Records: 12 Duplicates: 0 Warnings: 0
```

```
MariaDB [hosakbari_db]> (17, NULL, '2044-07-03', 'geplant', 2, 'Elektro-LKW', 17.89), ^C
MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
1	Berlin, Alexanderplatz	2044-06-15	unterwegs	NULL	NULL	NULL
2	Hamburg, Hafenstraße	2044-06-16	geliefert	NULL	NULL	NULL
3	München, Marienplatz	2044-06-17	verspätet	NULL	NULL	NULL
4	Frankfurt, Zeilstraße	2044-06-18	geplant	NULL	NULL	NULL
5	Unbekannt	2044-06-20	in Bearbeitung	NULL	NULL	NULL
6	Unbekannt	2044-06-21	Verspätet	NULL	NULL	NULL
7	Unbekannt	2044-06-22	Verspätet	NULL	NULL	NULL
8	Unbekannt	2044-06-23	Verspätet	NULL	NULL	NULL
9	Köln, Domplatz	2044-06-25	geplant	2	Drohne	19.99
10	NULL	2044-06-27	unterwegs	1	Elektro-LKW	29.99
11	Hamburg, Reeperbahn	NULL	verspätet	3	Hybrid-LKW	24.50
12	München, Stachus	2044-07-01	geliefert	2	Elektro-Van	15.75
13	NULL	NULL	in Bearbeitung	1	Drohne	12.99
14	Düsseldorf, Altstadt	2044-07-05	NULL	2	Wasserstoff-LKW	22.30
15	Stuttgart, Schlossplatz	2044-06-29	unterwegs	1	Elektro-Scooter	9.99
16	Frankfurt, Mainufer	NULL	NULL	3	Drohne	14.50
17	NULL	2044-07-03	geplant	2	Elektro-LKW	17.89
18	Berlin, Kudamm	NULL	in Bearbeitung	1	Hybrid-Drohne	11.99
19	NULL	NULL	NULL	3	Autonomer LKW	31.50
20	Nürnberg, Zentrum	2044-07-10	verspätet	1	Drohne	10.99

20 rows in set (0.001 sec)

Jetzt führen wir die ursprünglichen Befehle erneut durch.

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung WHERE Zielort IS NULL;
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
10	NULL	2044-06-27	unterwegs	1	Elektro-LKW	29.99
13	NULL	NULL	in Bearbeitung	1	Drohne	12.99
17	NULL	2044-07-03	geplant	2	Elektro-LKW	17.89
19	NULL	NULL	NULL	3	Autonomer LKW	31.50

4 rows in set (0.001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung WHERE Lieferzeitpunkt IS NULL;
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
11	Hamburg, Reeperbahn	NULL	verspätet	3	Hybrid-LKW	24.50
13	NULL	NULL	in Bearbeitung	1	Drohne	12.99
16	Frankfurt, Mainufer	NULL	NULL	3	Drohne	14.50
18	Berlin, Kudamm	NULL	in Bearbeitung	1	Hybrid-Drohne	11.99
19	NULL	NULL	NULL	3	Autonomer LKW	31.50

5 rows in set (0.001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung WHERE Status IS NULL;
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
14	Düsseldorf, Altstadt	2044-07-05	NULL	2	Wasserstoff-LKW	22.30
16	Frankfurt, Mainufer	NULL	NULL	3	Drohne	14.50
19	NULL	NULL	NULL	3	Autonomer LKW	31.50

3 rows in set (0.001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung WHERE Lieferzeitpunkt > '2044-06-30';
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
12	München, Stachus	2044-07-01	geliefert	2	Elektro-Van	15.75
14	Düsseldorf, Altstadt	2044-07-05	NULL	2	Wasserstoff-LKW	22.30
17	NULL	2044-07-03	geplant	2	Elektro-LKW	17.89
20	Nürnberg, Zentrum	2044-07-10	verspätet	1	Drohne	10.99

4 rows in set (0.001 sec)

Relationen mit NULL-Werten für Lieferung

Dito für Lieferung

Es wird eine Abfrage ausgeführt, um alle Fahrzeuge anzuzeigen, bei denen keine Lieferungsnummer angegeben wurde.

```
SELECT * FROM Fahrzeug WHERE Lieferung_Lieferungsnummer IS NULL;
```

Es wird eine Abfrage ausgeführt, um alle Fahrzeuge anzuzeigen, bei denen der Batteriestatus nicht angegeben wurde.

```
SELECT * FROM Fahrzeug WHERE Batteriestatus IS NULL;
```

Es wird eine Abfrage ausgeführt, um alle Fahrzeuge anzuzeigen, bei denen der Fahrzeugtyp nicht angegeben wurde.

```
SELECT * FROM Fahrzeug WHERE Typ IS NULL;
```

Es wird eine Abfrage ausgeführt, um alle Fahrzeuge anzuzeigen, bei denen die Kapazität nicht angegeben wurde.

```
SELECT * FROM Fahrzeug WHERE Kapazität IS NULL;
```

Wir füllen die Tabelle Fahrzeug erstmal wieder mit Testdaten/Beispieldaten:

Es werden neue Fahrzeugdaten in die Tabelle Fahrzeug eingefügt. Die Werte umfassen Fahrzeug-IDs, Typen, Kapazitäten, Batteriestatus und zugehörige Lieferungsnummern. Beispielsweise wird ein Elektro-LKW mit einer Kapazität von 7000 und einem Batteriestatus von 90 hinzugefügt, ohne eine zugewiesene Lieferung.

```
INSERT INTO Fahrzeug (Fahrzeug_ID, Typ, Kapazität, Batteriestatus, Lieferung_Lieferungsnummer)
VALUES
(9, 'Elektro-LKW', 7000.0, 90, NULL),
(10, NULL, 1000.0, 100, 1),
(11, 'Hybrid-LKW', NULL, 80, 3),
(12, 'Drohne', 50.0, NULL, 4),
(13, 'Elektro-Van', 2000.0, 60, NULL),
(14, NULL, NULL, NULL, NULL),
(15, 'Autonomer LKW', 7500.0, 100, 2),
(16, 'Wasserstoff-LKW', NULL, NULL, 5),
(17, NULL, 8000.0, 75, NULL),
(18, 'Hybrid-Drohne', 85.0, 95, NULL);
```

```

MariaDB [hosakbari_db]> SELECT * FROM Lieferung WHERE Lieferzeitpunkt > '2044-06-30';^C
MariaDB [hosakbari_db]> INSERT INTO Fahrzeug (Fahrzeug_ID, Typ, Kapazität, Batteriestatus, Lieferung_Lieferungsnummer)
    → VALUES
    → (9, 'Elektro-LKW', 7000.0, 90, NULL),
    → (10, NULL, 1000.0, 100, 1),
    → (11, 'Hybrid-LKW', NULL, 80, 3),
    → (12, 'Drohne', 50.0, NULL, 4),
    → (13, 'Elektro-Van', 2000.0, 60, NULL),
    → (14, NULL, NULL, NULL, NULL),
    → (15, 'Autonomer LKW', 7500.0, 100, 2),
    → (16, 'Wasserstoff-LKW', NULL, NULL, 5),
    → (17, NULL, 8000.0, 75, NULL),
    → (18, 'Hybrid-Drohne', 85.0, 95, NULL);
Query OK, 10 rows affected (0,002 sec)
Records: 10 Duplicates: 0 Warnings: 0

```

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
```

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer	Wartungsstatus	Ladezeit	CO2_Einsparung
1	Elektro-LKW	5000	85		1	NULL	NULL
3	Elektro-Van	1100	75		3	NULL	NULL
5	Autonomer LKW	7000	100		3	NULL	NULL
6	Wasserstoff-LKW	6500	100		3	NULL	NULL
9	Elektro-LKW	7000	90		NULL	NULL	NULL
10	NULL	1000	100		1	NULL	NULL
11	Hybrid-LKW	NULL	80		3	NULL	NULL
12	Drohne	50	NULL		4	NULL	NULL
13	Elektro-Van	2000	60		NULL	NULL	NULL
14	NULL	NULL	NULL		NULL	NULL	NULL
15	Autonomer LKW	7500	100		2	NULL	NULL
16	Wasserstoff-LKW	NULL	NULL		5	NULL	NULL
17	NULL	8000	75		NULL	NULL	NULL
18	Hybrid-Drohne	85	95		NULL	NULL	NULL

14 rows in set (0,001 sec)

Nun führen wir die ursprünglichen Befehle durch.

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug WHERE Lieferung_Lieferungsnummer IS NULL;
```

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer	Wartungsstatus	Ladezeit	CO2_Einsparung
9	Elektro-LKW	7000	90		NULL	NULL	NULL
13	Elektro-Van	2000	60		NULL	NULL	NULL
14	NULL	NULL	NULL		NULL	NULL	NULL
17	NULL	8000	75		NULL	NULL	NULL
18	Hybrid-Drohne	85	95		NULL	NULL	NULL

5 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug WHERE Batteriestatus IS NULL;
```

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer	Wartungsstatus	Ladezeit	CO2_Einsparung
12	Drohne	50	NULL		4	NULL	NULL
14	NULL	NULL	NULL		NULL	NULL	NULL
16	Wasserstoff-LKW	NULL	NULL		5	NULL	NULL

3 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug WHERE Typ IS NULL;
```

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer	Wartungsstatus	Ladezeit	CO2_Einsparung
10	NULL	1000	100		1	NULL	NULL
14	NULL	NULL	NULL		NULL	NULL	NULL
17	NULL	8000	75		NULL	NULL	NULL

3 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug WHERE Kapazität IS NULL;
```

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer	Wartungsstatus	Ladezeit	CO2_Einsparung
11	Hybrid-LKW	NULL	80		3	NULL	NULL
14	NULL	NULL	NULL		NULL	NULL	NULL
16	Wasserstoff-LKW	NULL	NULL		5	NULL	NULL

3 rows in set (0,001 sec)

Relationen mit NULL-Werten für Lagerhaus

Dito für Lagerhaus

Es wird eine Abfrage ausgeführt, um alle Lagerhäuser anzuzeigen, bei denen die Energiequelle nicht angegeben wurde.

```
SELECT * FROM Lagerhaus WHERE Energiequelle IS NULL;
```

Es wird eine Abfrage ausgeführt, um alle Lagerhäuser anzuzeigen, bei denen der Standort nicht angegeben wurde.

```
SELECT * FROM Lagerhaus WHERE Standort IS NULL;
```

Es wird eine Abfrage ausgeführt, um alle Lagerhäuser anzuzeigen, bei denen die Kapazität nicht angegeben wurde.

```
SELECT * FROM Lagerhaus WHERE Kapazität IS NULL;
```

Es wird eine Abfrage ausgeführt, um alle Lagerhäuser anzuzeigen, bei denen die Sicherheitsstufe nicht angegeben wurde

```
SELECT * FROM Lagerhaus WHERE Sicherheitsstufe IS NULL;
```

Wir füllen die Tabelle wieder erst mit Testdaten auf, danach wenden wir die Befehle oben an:

Es werden neue Lagerhausdaten in die Tabelle Lagerhaus eingefügt. Die Werte umfassen Lager-IDs, Standorte, Kapazitäten, Energiequellen und Sicherheitsstufen. Beispielsweise wird ein Zentrallager in Leipzig mit einer Kapazität von 40000 und ohne definierte Energiequelle, aber mit einer hohen Sicherheitsstufe hinzugefügt.

```
INSERT INTO Lagerhaus (Lager_ID, Standort, Kapazität, Energiequelle, Sicherheitsstufe)
```

```
VALUES
```

```
(5, 'Leipzig, Zentrallager', 40000, NULL, 'Hoch'),  
(6, NULL, 60000, 'Solar', 'Mittel'),  
(7, 'Dortmund, Hafenlager', NULL, 'Wind', NULL),  
(8, 'Stuttgart, Logistikzentrum', 75000, NULL, 'Niedrig'),  
(9, NULL, NULL, 'Solar/Wind', NULL),  
(10, 'Düsseldorf, Industriepark', 85000, 'Wind', NULL);
```

```
MariaDB [hosakbari_db]> INSERT INTO Lagerhaus (Lager_ID, Standort, Kapazität, Energiequelle, Sicherheitsstufe)
```

```
→ VALUES  
→ (5, 'Leipzig, Zentrallager', 40000, NULL, 'Hoch'),  
→ (6, NULL, 60000, 'Solar', 'Mittel'),  
→ (7, 'Dortmund, Hafenlager', NULL, 'Wind', NULL),  
→ (8, 'Stuttgart, Logistikzentrum', 75000, NULL, 'Niedrig'),  
→ (9, NULL, NULL, 'Solar/Wind', NULL),  
→ (10, 'Düsseldorf, Industriepark', 85000, 'Wind', NULL);
```

```
ERROR 1366 (22007): Incorrect integer value: 'Hoch' for column 'hosakbari_db'. 'Lagerhaus'. 'Sicherheitsstufe' at row 1
```

```
MariaDB [hosakbari_db]> INSERT INTO Lagerhaus (Lager_ID, Standort, Kapazität, Energiequelle, Sicherheitsstufe) VALUES (5, 'Leipzig', 40000, 'Solar', 'Hoch'),  
(6, NULL, 60000, 'Solar', 'Mittel'),  
(7, 'Dortmund, Hafenlager', 35000, 'Wind', 'Niedrig'),  
(8, 'Stuttgart, Logistikzentrum', 75000, NULL, 'Niedrig'), (9, NULL, NULL, 'Solar/Wind', NULL), (10, 'Düss')
```

```
ERROR 1366 (22007): Incorrect integer value: 'Hoch' for column 'hosakbari_db'. 'Lagerhaus'. 'Sicherheitsstufe' at row 1
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
```

Lager_ID	Standort	Kapazität	Energiequelle	Sicherheitsstufe	Kühlkapazität	Drohnenlandeplatz
1	Berlin, Lagerstraße 12	50000	Solar		NULL	NULL
2	Hamburg, Hafenlager 23	75000	Wind		NULL	NULL
3	München, Südallee 45	35000	Solar		NULL	NULL
4	Frankfurt, Westhafen 8	100000	Solar		NULL	NULL
5	Leipzig, Zentrallager	45000	Erneuerbar		NULL	NULL
6	Stuttgart, Hauptlager 1	55000	Erneuerbar		NULL	NULL
7	Düsseldorf, Logistikzentrum	50000	Erneuerbar		NULL	NULL
8	Dresden, Verteilzentrum	60000	Erneuerbar		NULL	NULL

8 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus WHERE Energiequelle IS NULL;  
Empty set (0,001 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus WHERE Standort IS NULL;  
Empty set (0,001 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus WHERE Kapazität IS NULL;  
Empty set (0,001 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus WHERE Sicherheitsstufe IS NULL;
```

Lager_ID	Standort	Kapazität	Energiequelle	Sicherheitsstufe	Kühlkapazität	Drohnenlandeplatz
1	Berlin, Lagerstraße 12	50000	Solar		NULL	NULL
2	Hamburg, Hafenlager 23	75000	Wind		NULL	NULL
3	München, Südallee 45	35000	Solar		NULL	NULL
4	Frankfurt, Westhafen 8	100000	Solar		NULL	NULL
5	Leipzig, Zentrallager	45000	Erneuerbar		NULL	NULL
6	Stuttgart, Hauptlager 1	55000	Erneuerbar		NULL	NULL
7	Düsseldorf, Logistikzentrum	50000	Erneuerbar		NULL	NULL
8	Dresden, Verteilzentrum	60000	Erneuerbar		NULL	NULL

8 rows in set (0,001 sec)

Hierzu bekam ich nur Einträge für Letzteres. Egal.

Relationen mit NULL-Werten für Artikel

Dito für Artikel

Es wird eine Abfrage ausgeführt, um alle Artikel anzuzeigen, bei denen keine Lagerhaus-ID angegeben wurde.

```
SELECT * FROM Artikel WHERE Lagerhaus_Lager_ID IS NULL;
```

Es wird eine Abfrage ausgeführt, um alle Artikel anzuzeigen, bei denen die Bezeichnung nicht angegeben wurde.

```
SELECT * FROM Artikel WHERE Bezeichnung IS NULL;
```

Es wird eine Abfrage ausgeführt, um alle Artikel anzuzeigen, bei denen der Lagerbestand nicht angegeben wurde.

```
SELECT * FROM Artikel WHERE Lagerbestand IS NULL;
```

Es wird eine Abfrage ausgeführt, um alle Artikel anzuzeigen, bei denen das Gewicht nicht angegeben wurde.

```
SELECT * FROM Artikel WHERE Gewicht IS NULL;
```

Zuerst füllen wir die Tabelle Artikel mit Beispieldaten:

Es werden neue Artikeldaten in die Tabelle Artikel eingefügt. Die Werte umfassen Artikelnummern, Bezeichnungen, Gewichte, Lagerbestände und zugehörige Lagerhaus-IDs. Beispielsweise wird ein Drucker Modell T100 mit einem Gewicht von 5.0, einem Lagerbestand von 100 und keiner zugewiesenen Lagerhaus-ID hinzugefügt.

```
INSERT INTO Artikel (Artikelnummer, Bezeichnung, Gewicht, Lagerbestand, Lagerhaus_Lager_ID)
```

```
VALUES
```

```
(106, 'Drucker Modell T100', 5.0, 100, NULL),  
(107, NULL, 2.5, 50, 5),  
(108, 'Smartwatch Serie Z', NULL, 200, 6),  
(109, 'Gaming-Laptop Ultra', 3.5, NULL, 7),  
(110, 'Elektro-Kocher', 4.2, 80, NULL),  
(111, NULL, NULL, NULL, NULL);
```

```
MariaDB [hosakbari_db]> INSERT INTO Artikel (Artikelnummer, Bezeichnung, Gewicht, Lagerbestand, Lagerhaus_Lager_ID)  
    → VALUES  
    → (106, 'Drucker Modell T100', 5.0, 100, NULL),  
    → (107, NULL, 2.5, 50, 5),  
    → (108, 'Smartwatch Serie Z', NULL, 200, 6),  
    → (109, 'Gaming-Laptop Ultra', 3.5, NULL, 7),  
    → (110, 'Elektro-Kocher', 4.2, 80, NULL),  
    → (111, NULL, NULL, NULL, NULL);  
ERROR 1062 (23000): Duplicate entry '106' for key 'PRIMARY'  
MariaDB [hosakbari_db]> SELECT * FROM Artikel;
```

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID	Produktionsland	Garantiezeit	Recyclingklasse
101	Smartphone-Modell X	0.24	1450		1	NULL	NULL
102	Laptop-Modell Y	1.8	450	2	NULL	NULL	NULL
103	Tablet-Modell Z	0.96	750	3	NULL	NULL	NULL
104	Kühlschrank Eco-Kühl	50	70	4	NULL	NULL	NULL
105	Waschmaschine SuperClean	75	50	4	NULL	NULL	NULL
106	Drucker Modell T100	5	100	1	NULL	NULL	NULL
107	3D-Drucker HighTech	10	150	1	NULL	NULL	NULL
108	Neues Produkt	0.6	250	1	NULL	NULL	NULL
109	Smartwatch Pro	0.36	450	1	NULL	NULL	NULL

9 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Artikel WHERE Lagerhaus_Lager_ID IS NULL;  
Empty set (0,002 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Artikel WHERE Bezeichnung IS NULL;  
Empty set (0,001 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Artikel WHERE Lagerbestand IS NULL;  
Empty set (0,001 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Artikel WHERE Gewicht IS NULL;  
Empty set (0,001 sec)
```

Hierfür bekam ich auch keine Einträge.

Es ist in SQL ebenfalls möglich SQL-Operationen einzufügen, wobei Nullwerte automatisch in die Tabelle eingefügt werden, ohne dass explizit NULL verwendet wird. Das erreicht man, indem man keine Werte für bestimmte Spalten angibt, vorausgesetzt, dass diese Spalten auf NULL als Standardwert gesetzt sind. Es folgt ...

Einfügen von Nullwerten für Lieferung

Es wird ein neuer Eintrag in die Tabelle Lieferung eingefügt. Der Eintrag fügt eine Lieferung mit der Nummer 21 hinzu, die in die Bremer Innenstadt am 12. Juli 2044 geliefert werden soll, wobei der Standardstatus verwendet wird

```
INSERT INTO Lieferung (Lieferungsnummer, Zielort, Lieferzeitpunkt, Status)
VALUES (21, 'Bremen, Innenstadt', '2044-07-12', DEFAULT);
```

Es wird ein neuer Eintrag in die Tabelle Lieferung eingefügt. Der Eintrag fügt eine Lieferung mit der Nummer 22 hinzu, bei der der Zielort als Standardwert und der Lieferzeitpunkt als 13. Juli 2044 gesetzt wird, mit dem Status geplant.

```
INSERT INTO Lieferung (Lieferungsnummer, Zielort, Lieferzeitpunkt, Status)
VALUES (22, DEFAULT, '2044-07-13', 'geplant');
```

Es wird ein neuer Eintrag in die Tabelle Lieferung eingefügt. Der Eintrag fügt eine Lieferung mit der Nummer 23 hinzu, die zum Hauptbahnhof Hannover gehen soll, mit einem Standardwert für den Lieferzeitpunkt und dem Status verspätet.

```
INSERT INTO Lieferung (Lieferungsnummer, Zielort, Lieferzeitpunkt, Status)
VALUES (23, 'Hannover, Hauptbahnhof', DEFAULT, 'verspätet');
```

Es wird ein neuer Eintrag in die Tabelle Lieferung eingefügt. Der Eintrag fügt eine Lieferung mit der Nummer 24 hinzu, bei der alle Werte als Standard gesetzt werden.

```
INSERT INTO Lieferung (Lieferungsnummer, Zielort, Lieferzeitpunkt, Status)
VALUES (24, DEFAULT, DEFAULT, DEFAULT);
```

Hier wird zum Beispiel durch DEFAULT dafür gesorgt, dass Nullwerte automatisch eingefügt werden, wenn die Spalte standardmäßig NULL erlaubt. Wenn keine Werte für einer Spalte angegeben werden, wird NULL eingetragen.

```
MariaDB [hosakbari_db]> INSERT INTO Lieferung (Lieferungsnummer, Zielort, Lieferzeitpunkt, Status)
-> VALUES (21, 'Bremen, Innenstadt', '2044-07-12', DEFAULT);
Query OK, 1 row affected (0.020 sec)

MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
+-----+-----+-----+-----+-----+-----+-----+
| Lieferungsnummer | Zielort | Lieferzeitpunkt | Status | Priorität | Transporttyp | Versandkosten |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Berlin, Alexanderplatz | 2044-06-15 | unterwegs | NULL | NULL | NULL |
| 2 | Hamburg, Hafenstraße | 2044-06-16 | geliefert | NULL | NULL | NULL |
| 3 | München, Marienplatz | 2044-06-17 | verspätet | NULL | NULL | NULL |
| 4 | Frankfurt, Zeilstraße | 2044-06-18 | geplant | NULL | NULL | NULL |
| 5 | Unbekannt | 2044-06-20 | in Bearbeitung | NULL | NULL | NULL |
| 6 | Unbekannt | 2044-06-21 | Verspätet | NULL | NULL | NULL |
| 7 | Unbekannt | 2044-06-22 | Verspätet | NULL | NULL | NULL |
| 8 | Unbekannt | 2044-06-23 | Verspätet | NULL | NULL | NULL |
| 9 | Köln, Domplatz | 2044-06-25 | geplant | 2 | Drohne | 19.99 |
| 10 | NULL | 2044-06-27 | unterwegs | 1 | Elektro-LKW | 29.99 |
| 11 | Hamburg, Reeperbahn | NULL | verspätet | 3 | Hybrid-LKW | 24.50 |
| 12 | München, Stachus | 2044-07-01 | geliefert | 2 | Elektro-Van | 15.75 |
| 13 | NULL | NULL | in Bearbeitung | 1 | Drohne | 12.99 |
| 14 | Düsseldorf, Altstadt | 2044-07-05 | NULL | 2 | Wasserstoff-LKW | 22.30 |
| 15 | Stuttgart, Schlossplatz | 2044-06-29 | unterwegs | 1 | Elektro-Scooter | 9.99 |
| 16 | Frankfurt, Mainufer | NULL | NULL | 3 | Drohne | 14.50 |
| 17 | NULL | 2044-07-03 | geplant | 2 | Elektro-LKW | 17.89 |
| 18 | Berlin, Kudamm | NULL | in Bearbeitung | 1 | Hybrid-Drohne | 11.99 |
| 19 | NULL | NULL | NULL | 3 | Autonomer LKW | 31.50 |
| 20 | Nürnberg, Zentrum | 2044-07-10 | verspätet | 1 | Drohne | 10.99 |
| 21 | Bremen, Innenstadt | 2044-07-12 | NULL | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+
21 rows in set (0.001 sec)
```

```
MariaDB [hosakbari_db]> INSERT INTO Lieferung (Lieferungsnummer, Zielort, Lieferzeitpunkt, Status)
'2044-07-13', 'g' → VALUES (22, DEFAULT, '2044-07-13', 'geplant');
Query OK, 1 row affected (0,002 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
1	Berlin, Alexanderplatz	2044-06-15	unterwegs	NULL	NULL	NULL
2	Hamburg, Hafenstraße	2044-06-16	geliefert	NULL	NULL	NULL
3	München, Marienplatz	2044-06-17	verspätet	NULL	NULL	NULL
4	Frankfurt, Zeilstraße	2044-06-18	geplant	NULL	NULL	NULL
5	Unbekannt	2044-06-20	in Bearbeitung	NULL	NULL	NULL
6	Unbekannt	2044-06-21	Verspätet	NULL	NULL	NULL
7	Unbekannt	2044-06-22	Verspätet	NULL	NULL	NULL
8	Unbekannt	2044-06-23	Verspätet	NULL	NULL	NULL
9	Köln, Domplatz	2044-06-25	geplant	2	Drohne	19.99
10	NULL	2044-06-27	unterwegs	1	Elektro-LKW	29.99
11	Hamburg, Reeperbahn	NULL	verspätet	3	Hybrid-LKW	24.50
12	München, Stachus	2044-07-01	geliefert	2	Elektro-Van	15.75
13	NULL	NULL	in Bearbeitung	1	Drohne	12.99
14	Düsseldorf, Altstadt	2044-07-05	NULL	2	Wasserstoff-LKW	22.30
15	Stuttgart, Schlossplatz	2044-06-29	unterwegs	1	Elektro-Scooter	9.99
16	Frankfurt, Mainufer	NULL	NULL	3	Drohne	14.50
17	NULL	2044-07-03	geplant	2	Elektro-LKW	17.89
18	Berlin, Kudamm	NULL	in Bearbeitung	1	Hybrid-Drohne	11.99
19	NULL	NULL	NULL	3	Autonomer LKW	31.50
20	Nürnberg, Zentrum	2044-07-10	verspätet	1	Drohne	10.99
21	Bremen, Innenstadt	2044-07-12	NULL	NULL	NULL	NULL
22	NULL	2044-07-13	geplant	NULL	NULL	NULL

22 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> INSERT INTO Lieferung (Lieferungsnummer, Zielort, Lieferzeitpunkt, Status)
→ VALUES (23, 'Hannover, Hauptbahnhof', DEFAULT, 'verspätet');
Query OK, 1 row affected (0,002 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
1	Berlin, Alexanderplatz	2044-06-15	unterwegs	NULL	NULL	NULL
2	Hamburg, Hafenstraße	2044-06-16	geliefert	NULL	NULL	NULL
3	München, Marienplatz	2044-06-17	verspätet	NULL	NULL	NULL
4	Frankfurt, Zeilstraße	2044-06-18	geplant	NULL	NULL	NULL
5	Unbekannt	2044-06-20	in Bearbeitung	NULL	NULL	NULL
6	Unbekannt	2044-06-21	Verspätet	NULL	NULL	NULL
7	Unbekannt	2044-06-22	Verspätet	NULL	NULL	NULL
8	Unbekannt	2044-06-23	Verspätet	NULL	NULL	NULL
9	Köln, Domplatz	2044-06-25	geplant	2	Drohne	19.99
10	NULL	2044-06-27	unterwegs	1	Elektro-LKW	29.99
11	Hamburg, Reeperbahn	NULL	verspätet	3	Hybrid-LKW	24.50
12	München, Stachus	2044-07-01	geliefert	2	Elektro-Van	15.75
13	NULL	NULL	in Bearbeitung	1	Drohne	12.99
14	Düsseldorf, Altstadt	2044-07-05	NULL	2	Wasserstoff-LKW	22.30
15	Stuttgart, Schlossplatz	2044-06-29	unterwegs	1	Elektro-Scooter	9.99
16	Frankfurt, Mainufer	NULL	NULL	3	Drohne	14.50
17	NULL	2044-07-03	geplant	2	Elektro-LKW	17.89
18	Berlin, Kudamm	NULL	in Bearbeitung	1	Hybrid-Drohne	11.99
19	NULL	NULL	NULL	3	Autonomer LKW	31.50
20	Nürnberg, Zentrum	2044-07-10	verspätet	1	Drohne	10.99
21	Bremen, Innenstadt	2044-07-12	NULL	NULL	NULL	NULL
22	NULL	2044-07-13	geplant	NULL	NULL	NULL
23	Hannover, Hauptbahnhof	NULL	verspätet	NULL	NULL	NULL

23 rows in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT * FROM Lieferung;						
Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
1	Berlin, Alexanderplatz	2044-06-15	unterwegs	NULL	NULL	NULL
2	Hamburg, Hafenstraße	2044-06-16	geliefert	NULL	NULL	NULL
3	München, Marienplatz	2044-06-17	verspätet	NULL	NULL	NULL
4	Frankfurt, Zeilstraße	2044-06-18	geplant	NULL	NULL	NULL
5	Unbekannt	2044-06-20	in Bearbeitung	NULL	NULL	NULL
6	Unbekannt	2044-06-21	Verspätet	NULL	NULL	NULL
7	Unbekannt	2044-06-22	Verspätet	NULL	NULL	NULL
8	Unbekannt	2044-06-23	Verspätet	NULL	NULL	NULL
9	Köln, Domplatz	2044-06-25	geplant	2	Drohne	19.99
10	NULL	2044-06-27	unterwegs	1	Elektro-LKW	29.99
11	Hamburg, Reeperbahn	NULL	verspätet	3	Hybrid-LKW	24.50
12	München, Stachus	2044-07-01	geliefert	2	Elektro-Van	15.75
13	NULL	NULL	in Bearbeitung	1	Drohne	12.99
14	Düsseldorf, Altstadt	2044-07-05	NULL	2	Wasserstoff-LKW	22.30
15	Stuttgart, Schlossplatz	2044-06-29	unterwegs	1	Elektro-Scooter	9.99
16	Frankfurt, Mainufer	NULL	NULL	3	Drohne	14.50
17	NULL	2044-07-03	geplant	2	Elektro-LKW	17.89
18	Berlin, Kudamm	NULL	in Bearbeitung	1	Hybrid-Drohne	11.99
19	NULL	NULL	NULL	3	Autonomer LKW	31.50
20	Nürnberg, Zentrum	2044-07-10	verspätet	1	Drohne	10.99
21	Bremen, Innenstadt	2044-07-12	NULL	NULL	NULL	NULL
22	NULL	2044-07-13	geplant	NULL	NULL	NULL
23	Hannover, Hauptbahnhof	NULL	verspätet	NULL	NULL	NULL
24	NULL	NULL	NULL	NULL	NULL	NULL

24 rows in set (0,001 sec)

Dasselbe könnte man mit den Tabellen Fahrzeug, Lagerhaus, Artikel, sowie Relation_Lieferung_Artikel machen.
Aber das überspringe ich.

Aggregatfunktionen: COUNT(*), Anzahl der Tupel; COUNT(attr), Anzahl der Tupel ohne Null für attr; COUNT(DISTINCT attr), Anzahl der verschiedenen Tupel ohne Null für attr; SUM(attr), Summe der Werte von attr (nur für numerische Attribute); AVG(attr), Durchschnitt der Werte von attr (nur für numerische Attribute), MIN(attr), MAX(attr); Kombination von GROUP BY, ORDER BY; HAVING, Aggregatfunktionen; Problematik „Logistik in 20 Jahren“

Aggregatfunktionen – Erklärung

Bevor ich die Aufgabenstellung mache, sollte geklärt werden was unter Aggregatfunktionen zu verstehen ist: Aggregatfunktionen berechnen Werte über eine ganze Spalte. Durch sie wird ein einzelner Wert zurückgeliefert, anstatt mehrere Zeilen anzuzeigen. Des Weiteren werden sie des Öfteren mit GROUP BY, ORDER BY und HAVING kombiniert, um Statistiken aus der Datenbank zu erhalten. COUNT(*) zum Beispiel zählt alle Zeilen (inklusive NULL). COUNT(attr) wiederum zählt Zeilen mit nicht-null Werten. COUNT(DISTINCT attr) zählt eindeutige Werte (ohne NULL). Durch SUM(attr) wird die Gesamtsumme einer numerischen Spalte berechnet. Nachfolgend wird durch AVG(attr) der Durchschnitt einer numerischen Spalte berechnet. Letztlich wird durch MIN(attr) und MAX(attr) der kleinste bzw. größte Wert geliefert.

Aggregatfunktion für Lieferung

Unten werden durch COUNT(*) alle Lieferungen gezählt. COUNT(Zielort) zählt alle Lieferungen mit bekannten Zielort (ohne NULL). COUNT(DISTINCT Zielort) gibt die Anzahl verschiedener Zielorte zurück. SUM(Versandkosten) berechnet die Gesamtkosten aller Lieferungen. AVG(Versandkosten) gibt die durchschnittlichen Versandkosten zurück. MIN(Lieferzeitpunkt), MAX(Lieferzeitpunkt) bestimmen die erste und letzte Lieferung.

Es wird eine Abfrage ausgeführt, um die Gesamtanzahl der Datensätze in der Tabelle Lieferung zu zählen.

```
SELECT COUNT(*) FROM Lieferung;
```

Es wird eine Abfrage ausgeführt, um die Anzahl der nicht-null Zielorte in der Tabelle Lieferung zu zählen.

```
SELECT COUNT(Zielort) FROM Lieferung;
```

Es wird eine Abfrage ausgeführt, um die Anzahl der unterschiedlichen Zielorte in der Tabelle Lieferung zu zählen.

```
SELECT COUNT(DISTINCT Zielort) FROM Lieferung;
```

Es wird eine Abfrage ausgeführt, um die Summe aller Versandkosten in der Tabelle Lieferung zu berechnen.

```
SELECT SUM(Versandkosten) FROM Lieferung;
```

Es wird eine Abfrage ausgeführt, um den Durchschnitt der Versandkosten in der Tabelle Lieferung zu berechnen.

```
SELECT AVG(Versandkosten) FROM Lieferung;
```

Es wird eine Abfrage ausgeführt, um das früheste Lieferdatum in der Tabelle Lieferung zu ermitteln.

```
SELECT MIN(Lieferzeitpunkt) FROM Lieferung;
```

Es wird eine Abfrage ausgeführt, um das späteste Lieferdatum in der Tabelle Lieferung zu ermitteln.

```
SELECT MAX(Lieferzeitpunkt) FROM Lieferung;
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
1	Berlin, Alexanderplatz	2044-06-15	unterwegs	NULL	NULL	NULL
2	Hamburg, Hafenstraße	2044-06-16	geliefert	NULL	NULL	NULL
3	München, Marienplatz	2044-06-17	verspätet	NULL	NULL	NULL
4	Frankfurt, Zeilstraße	2044-06-18	geplant	NULL	NULL	NULL
5	Unbekannt	2044-06-20	in Bearbeitung	NULL	NULL	NULL
6	Unbekannt	2044-06-21	Verspätet	NULL	NULL	NULL
7	Unbekannt	2044-06-22	Verspätet	NULL	NULL	NULL
8	Unbekannt	2044-06-23	Verspätet	NULL	NULL	NULL
9	Köln, Domplatz	2044-06-25	geplant	2	Drohne	19.99
10	NULL	2044-06-27	unterwegs	1	Elektro-LKW	29.99
11	Hamburg, Reeperbahn	NULL	verspätet	3	Hybrid-LKW	24.50
12	München, Stachus	2044-07-01	geliefert	2	Elektro-Van	15.75
13	NULL	NULL	in Bearbeitung	1	Drohne	12.99
14	Düsseldorf, Altstadt	2044-07-05	NULL	2	Wasserstoff-LKW	22.30
15	Stuttgart, Schlossplatz	2044-06-29	unterwegs	1	Elektro-Scooter	9.99
16	Frankfurt, Mainufer	NULL	NULL	3	Drohne	14.50
17	NULL	2044-07-03	geplant	2	Elektro-LKW	17.89
18	Berlin, Kudamm	NULL	in Bearbeitung	1	Hybrid-Drohne	11.99
19	NULL	NULL	NULL	3	Autonomer LKW	31.50
20	Nürnberg, Zentrum	2044-07-10	verspätet	1	Drohne	10.99
21	Bremen, Innenstadt	2044-07-12	NULL	NULL	NULL	NULL
22	NULL	2044-07-13	geplant	NULL	NULL	NULL
23	Hannover, Hauptbahnhof	NULL	verspätet	NULL	NULL	NULL
24	NULL	NULL	NULL	NULL	NULL	NULL

24 rows in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT COUNT(*) FROM Lieferung;

COUNT(*)
24

1 row in set (0,021 sec)

MariaDB [hosakbari_db]> SELECT COUNT(Zielort) FROM Lieferung;

COUNT(Zielort)
18

1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT COUNT(DISTINCT Zielort) FROM Lieferung
→ ;

COUNT(DISTINCT Zielort)
15

1 row in set (0,002 sec)

MariaDB [hosakbari_db]> SELECT SUM(Versandkosten) FROM Lieferung;

SUM(Versandkosten)
222.38

1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT AVG(Versandkosten) FROM Lieferung;

AVG(Versandkosten)
18.531667

1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT MIN(Lieferzeitpunkt) FROM Lieferung;

MIN(Lieferzeitpunkt)
2044-06-15

1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT MAX(Lieferzeitpunkt) FROM Lieferung;

MAX(Lieferzeitpunkt)
2044-07-13

1 row in set (0,001 sec)

Aggregatfunktion für Fahrzeug

Dito für Fahrzeug

Es wird eine Abfrage ausgeführt, um die Gesamtanzahl der Datensätze in der Tabelle Fahrzeug zu zählen.

```
SELECT COUNT(*) FROM Fahrzeug;
```

Es wird eine Abfrage ausgeführt, um die Anzahl der nicht-null Fahrzeugtypen in der Tabelle Fahrzeug zu zählen.

```
SELECT COUNT(Typ) FROM Fahrzeug;
```

Es wird eine Abfrage ausgeführt, um die Anzahl der unterschiedlichen Fahrzeugtypen in der Tabelle Fahrzeug zu zählen.

```
SELECT COUNT(DISTINCT Typ) FROM Fahrzeug;
```

Es wird eine Abfrage ausgeführt, um die Gesamtkapazität aller Fahrzeuge in der Tabelle Fahrzeug zu berechnen.

```
SELECT SUM(Kapazität) FROM Fahrzeug;
```

Es wird eine Abfrage ausgeführt, um den durchschnittlichen Batteriestatus aller Fahrzeuge in der Tabelle Fahrzeug zu berechnen.

```
SELECT AVG(Batteriestatus) FROM Fahrzeug;
```

Es wird eine Abfrage ausgeführt, um die kleinste Kapazität eines Fahrzeugs in der Tabelle Fahrzeug zu ermitteln.

```
SELECT MIN(Kapazität) FROM Fahrzeug;
```

Es wird eine Abfrage ausgeführt, um die größte Kapazität eines Fahrzeugs in der Tabelle Fahrzeug zu ermitteln.

```
SELECT MAX(Kapazität) FROM Fahrzeug;
```

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
```

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer	Wartungsstatus	Ladezeit	CO2_Einsparung
1	Elektro-LKW	5000	85		1	NULL	NULL
3	Elektro-Van	1100	75		3	NULL	NULL
5	Autonomer LKW	7000	100		3	NULL	NULL
6	Wasserstoff-LKW	6500	100		3	NULL	NULL
9	Elektro-LKW	7000	90		NULL	NULL	NULL
10	NULL	1000	100		1	NULL	NULL
11	Hybrid-LKW	NULL	80		3	NULL	NULL
12	Drohne	50	NULL		4	NULL	NULL
13	Elektro-Van	2000	60		NULL	NULL	NULL
14	NULL	NULL	NULL		NULL	NULL	NULL
15	Autonomer LKW	7500	100		2	NULL	NULL
16	Wasserstoff-LKW	NULL	NULL		5	NULL	NULL
17	NULL	8000	75		NULL	NULL	NULL
18	Hybrid-Drohne	85	95		NULL	NULL	NULL

14 rows in set (0,014 sec)

```

MariaDB [hosakbari_db]> SELECT COUNT(*) FROM Fahrzeug;
+-----+
| COUNT(*) |
+-----+
|      14 |
+-----+
1 row in set (0,024 sec)

MariaDB [hosakbari_db]> SELECT COUNT(Typ) FROM Fahrzeug;
+-----+
| COUNT(Typ) |
+-----+
|       11 |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT COUNT(DISTINCT Typ) FROM Fahrzeug;
+-----+
| COUNT(DISTINCT Typ) |
+-----+
|         7 |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT SUM(Kapazität) FROM Fahrzeug;
+-----+
| SUM(Kapazität) |
+-----+
|     45235 |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT AVG(Batteriestatus) FROM Fahrzeug;
+-----+
| AVG(Batteriestatus) |
+-----+
|    87.2727 |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT MIN(Kapazität) FROM Fahrzeug;
+-----+
| MIN(Kapazität) |
+-----+
|       50 |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT MAX(Kapazität) FROM Fahrzeug;
+-----+
| MAX(Kapazität) |
+-----+
|     8000 |
+-----+
1 row in set (0,001 sec)

```

Aggregatfunktion für Lagerhaus

Dito für Lagerhaus

Es wird eine Abfrage ausgeführt, um die Gesamtanzahl der Datensätze in der Tabelle Lagerhaus zu zählen.

```
SELECT COUNT(*) FROM Lagerhaus;
```

Es wird eine Abfrage ausgeführt, um die Anzahl der nicht-null Energiequellen in der Tabelle Lagerhaus zu zählen.

```
SELECT COUNT(Energiequelle) FROM Lagerhaus;
```

Es wird eine Abfrage ausgeführt, um die Anzahl der unterschiedlichen Energiequellen in der Tabelle Lagerhaus zu zählen.

```
SELECT COUNT(DISTINCT Energiequelle) FROM Lagerhaus;
```

Es wird eine Abfrage ausgeführt, um die Gesamtkapazität aller Lagerhäuser in der Tabelle Lagerhaus zu berechnen.

```
SELECT SUM(Kapazität) FROM Lagerhaus;
```

Es wird eine Abfrage ausgeführt, um die durchschnittliche Kapazität aller Lagerhäuser in der Tabelle Lagerhaus zu berechnen.

```
SELECT AVG(Kapazität) FROM Lagerhaus;
```

Es wird eine Abfrage ausgeführt, um die kleinste Kapazität eines Lagerhauses in der Tabelle Lagerhaus zu ermitteln.

```
SELECT MIN(Kapazität) FROM Lagerhaus;
```

Es wird eine Abfrage ausgeführt, um die größte Kapazität eines Lagerhauses in der Tabelle Lagerhaus zu ermitteln.

```
SELECT MAX(Kapazität) FROM Lagerhaus;
```

MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;						
Lager_ID	Standort	Kapazität	Energiequelle	Sicherheitsstufe	Kühlkapazität	Drohnenlandeplatz
1	Berlin, Lagerstraße 12	50000	Solar	NULL	NULL	NULL
2	Hamburg, Hafenlager 23	75000	Wind	NULL	NULL	NULL
3	München, Südallee 45	35000	Solar	NULL	NULL	NULL
4	Frankfurt, Westhafen 8	100000	Solar	NULL	NULL	NULL
5	Leipzig, Zentrallager	45000	Erneuerbar	NULL	NULL	NULL
6	Stuttgart, Hauptlager 1	55000	Erneuerbar	NULL	NULL	NULL
7	Düsseldorf, Logistikzentrum	50000	Erneuerbar	NULL	NULL	NULL
8	Dresden, Verteilzentrum	60000	Erneuerbar	NULL	NULL	NULL

8 rows in set (0,012 sec)

```

MariaDB [hosakbari_db]> SELECT COUNT(*) FROM Lagerhaus;
+-----+
| COUNT(*) |
+-----+
|      8   |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT COUNT(Energiequelle) FROM Lagerhaus;
+-----+
| COUNT(Energiequelle) |
+-----+
|        8            |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT COUNT(DISTINCT Energiequelle) FROM Lagerhaus;
+-----+
| COUNT(DISTINCT Energiequelle) |
+-----+
|          3                |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT SUM(Kapazität) FROM Lagerhaus;
+-----+
| SUM(Kapazität) |
+-----+
|     470000    |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT AVG(Kapazität) FROM Lagerhaus;
+-----+
| AVG(Kapazität) |
+-----+
| 58750.0000    |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT MIN(Kapazität) FROM Lagerhaus;
+-----+
| MIN(Kapazität) |
+-----+
|      35000    |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT MAX(Kapazität) FROM Lagerhaus;
+-----+
| MAX(Kapazität) |
+-----+
|     100000    |
+-----+
1 row in set (0,001 sec)

```

Aggregatfunktion für Artikel

Dito für Artikel

Es wird eine Abfrage ausgeführt, um die Gesamtanzahl der Datensätze in der Tabelle Artikel zu zählen.

`SELECT COUNT(*) FROM Artikel;`

Es wird eine Abfrage ausgeführt, um die Anzahl der nicht-null Bezeichnungen in der Tabelle Artikel zu zählen.

`SELECT COUNT(Bezeichnung) FROM Artikel;`

Es wird eine Abfrage ausgeführt, um die Anzahl der unterschiedlichen Bezeichnungen in der Tabelle Artikel zu zählen.

`SELECT COUNT(DISTINCT Bezeichnung) FROM Artikel;`

Es wird eine Abfrage ausgeführt, um das Gesamtgewicht aller Artikel in der Tabelle Artikel zu berechnen.

`SELECT SUM(Gewicht) FROM Artikel;`

Es wird eine Abfrage ausgeführt, um das durchschnittliche Gewicht aller Artikel in der Tabelle Artikel zu berechnen.

```
SELECT AVG(Gewicht) FROM Artikel;
```

Es wird eine Abfrage ausgeführt, um das kleinste Gewicht eines Artikels in der Tabelle Artikel zu ermitteln.

```
SELECT MIN(Gewicht) FROM Artikel;
```

Es wird eine Abfrage ausgeführt, um das größte Gewicht eines Artikels in der Tabelle Artikel zu ermitteln.

```
SELECT MAX(Gewicht) FROM Artikel;
```

MariaDB [hosakbari_db]> SELECT * FROM Artikel;								
Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID	Produktionsland	Garantiezeit	Recyclingklasse	
101	Smartphone-Modell X	0.24	1450	1	NULL	NULL	NULL	
102	Laptop-Modell Y	1.8	450	2	NULL	NULL	NULL	
103	Tablet-Modell Z	0.96	750	3	NULL	NULL	NULL	
104	Kühlschrank Eco-Kühl	50	70	4	NULL	NULL	NULL	
105	Waschmaschine SuperClean	75	50	4	NULL	NULL	NULL	
106	Drucker Modell T100	5	100	1	NULL	NULL	NULL	
107	3D-Drucker HighTech	10	150	1	NULL	NULL	NULL	
108	Neues Produkt	0.6	250	1	NULL	NULL	NULL	
109	Smartwatch Pro	0.36	450	1	NULL	NULL	NULL	

```

MariaDB [hosakbari_db]> SELECT COUNT(*) FROM Artikel;
+-----+
| COUNT(*) |
+-----+
|      9   |
+-----+
1 row in set (0,012 sec)

MariaDB [hosakbari_db]> SELECT COUNT(Bezeichnung) FROM Artikel;
+-----+
| COUNT(Bezeichnung) |
+-----+
|          9         |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT COUNT(DISTINCT Bezeichnung) FROM Artikel;
+-----+
| COUNT(DISTINCT Bezeichnung) |
+-----+
|             9              |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT SUM(Gewicht) FROM Artikel;
+-----+
| SUM(Gewicht) |
+-----+
| 143.96000003814697 |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT AVG(Gewicht) FROM Artikel;
+-----+
| AVG(Gewicht) |
+-----+
| 15.995555559794107 |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT MIN(Gewicht) FROM Artikel;
+-----+
| MIN(Gewicht) |
+-----+
|      0.24    |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT MAX(Gewicht) FROM Artikel;
+-----+
| MAX(Gewicht) |
+-----+
|        75     |
+-----+
1 row in set (0,001 sec)

```

Aggregatfunktion für Relation_Lieferung_Artikel

Dito für Relation_Lieferung_Artikel

Es wird eine Abfrage ausgeführt, um die Gesamtanzahl der Datensätze in der Tabelle Relation_Lieferung_Artikel zu zählen.

```
SELECT COUNT(*) FROM Relation_Lieferung_Artikel;
```

Es wird eine Abfrage ausgeführt, um die Anzahl der nicht-null Liefermengen in der Tabelle Relation_Lieferung_Artikel zu zählen.

```
SELECT COUNT(Liefermenge) FROM Relation_Lieferung_Artikel;
```

Es wird eine Abfrage ausgeführt, um die Anzahl der unterschiedlichen Liefermengen in der Tabelle Relation_Lieferung_Artikel zu zählen.

```
SELECT COUNT(DISTINCT Liefermenge) FROM Relation_Lieferung_Artikel;
```

Es wird eine Abfrage ausgeführt, um die Gesamtliefermenge in der Tabelle Relation_Lieferung_Artikel zu berechnen.

```
SELECT SUM(Liefermenge) FROM Relation_Lieferung_Artikel;
```

Es wird eine Abfrage ausgeführt, um die durchschnittliche Liefermenge in der Tabelle Relation_Lieferung_Artikel zu berechnen.

```
SELECT AVG(Liefermenge) FROM Relation_Lieferung_Artikel;
```

Es wird eine Abfrage ausgeführt, um die kleinste Liefermenge in der Tabelle Relation_Lieferung_Artikel zu ermitteln.

```
SELECT MIN(Liefermenge) FROM Relation_Lieferung_Artikel;
```

Es wird eine Abfrage ausgeführt, um die größte Liefermenge in der Tabelle Relation_Lieferung_Artikel zu ermitteln.

```
SELECT MAX(Liefermenge) FROM Relation_Lieferung_Artikel;
```

Lieferungsnummer	Artikelnummer	Liefermenge	Versandart	Expressversand
1	101	NULL	NULL	NULL
2	103	NULL	NULL	NULL
4	101	NULL	NULL	NULL
4	103	NULL	NULL	NULL

4 rows in set (0,014 sec)

```

MariaDB [hosakbari_db]> SELECT COUNT(*) FROM Relation_Lieferung_Artikel;
+-----+
| COUNT(*) |
+-----+
|      4   |
+-----+
1 row in set (0,016 sec)

MariaDB [hosakbari_db]> SELECT COUNT(Liefermenge) FROM Relation_Lieferung_Artikel;
+-----+
| COUNT(Liefermenge) |
+-----+
|          0        |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT COUNT(DISTINCT Liefermenge) FROM Relation_Lieferung_Artikel;
+-----+
| COUNT(DISTINCT Liefermenge) |
+-----+
|             0              |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT SUM(Liefermenge) FROM Relation_Lieferung_Artikel;
+-----+
| SUM(Liefermenge) |
+-----+
|       NULL       |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT AVG(Liefermenge) FROM Relation_Lieferung_Artikel;
+-----+
| AVG(Liefermenge) |
+-----+
|       NULL       |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT MIN(Liefermenge) FROM Relation_Lieferung_Artikel;
+-----+
| MIN(Liefermenge) |
+-----+
|       NULL       |
+-----+
1 row in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT MAX(Liefermenge) FROM Relation_Lieferung_Artikel;
+-----+
| MAX(Liefermenge) |
+-----+
|       NULL       |
+-----+
1 row in set (0,001 sec)

```

Kombination von GROUP BY, ORDER BY, HAVING, Aggregatfunktionen

SQL erlaubt des Weiteren die Kombination von Aggregatfunktionen per GROUP BY, ORDER BY und HAVING, mit der man komplexe Analysen über Daten durchführen kann. Wir haben vorhin kennengelernt, dass Aggregatfunktionen Berechnungen auf die Gruppen anwenden kann, per COUNT, SUM, AVG, MIN, MAX. Mit GROUP BY kann man Zeilen mit gleichen Werten in einer oder mehreren Spalten gruppieren. Durch ORDER BY kann man die Ergebnisse nach einer Spalte oder einem berechneten Wert (ASC, DESC) sortieren. Durch HAVING werden Gruppen basierend auf Aggregatfunktionen (ähnlich zu WHERE, aber für Gruppen) gefiltert.

Es folgen Beispiele:

<pre> SELECT Status, COUNT(*) AS Anzahl_Lieferungen FROM Lieferung GROUP BY Status ORDER BY Anzahl_Lieferungen DESC; </pre>	Anzahl der Lieferungen pro Status
---	-----------------------------------

Es werden hierbei durch GROUP BY Status alle Lieferungen nach ihren aktuellen Status gruppiert. Per COUNT(*) AS Anzahl_Lieferungen wird gezählt, wie viele Lieferungen in jedem Status sind. Durch ORDER BY

Anzahl_Lieferungen DESC wird zuerst der Status mit den meisten Lieferungen gezeigt:

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
1	Berlin, Alexanderplatz	2044-06-15	unterwegs	NULL	NULL	NULL
2	Hamburg, Hafenstraße	2044-06-16	geliefert	NULL	NULL	NULL
3	München, Marienplatz	2044-06-17	verspätet	NULL	NULL	NULL
4	Frankfurt, Zeilstraße	2044-06-18	geplant	NULL	NULL	NULL
5	Unbekannt	2044-06-20	in Bearbeitung	NULL	NULL	NULL
6	Unbekannt	2044-06-21	Verspätet	NULL	NULL	NULL
7	Unbekannt	2044-06-22	Verspätet	NULL	NULL	NULL
8	Unbekannt	2044-06-23	Verspätet	NULL	NULL	NULL
9	Köln, Domplatz	2044-06-25	geplant	2	Drohne	19.99
10	NULL	2044-06-27	unterwegs	1	Elektro-LKW	29.99
11	Hamburg, Reeperbahn	NULL	verspätet	3	Hybrid-LKW	24.50
12	München, Stachus	2044-07-01	geliefert	2	Elektro-Van	15.75
13	NULL	NULL	in Bearbeitung	1	Drohne	12.99
14	Düsseldorf, Altstadt	2044-07-05	NULL	2	Wasserstoff-LKW	22.30
15	Stuttgart, Schlossplatz	2044-06-29	unterwegs	1	Elektro-Scooter	9.99
16	Frankfurt, Mainufer	NULL	NULL	3	Drohne	14.50
17	NULL	2044-07-03	geplant	2	Elektro-LKW	17.89
18	Berlin, Kudamm	NULL	in Bearbeitung	1	Hybrid-Drohne	11.99
19	NULL	NULL	NULL	3	Autonomer LKW	31.50
20	Nürnberg, Zentrum	2044-07-10	verspätet	1	Drohne	10.99
21	Bremen, Innenstadt	2044-07-12	NULL	NULL	NULL	NULL
22	NULL	2044-07-13	geplant	NULL	NULL	NULL
23	Hannover, Hauptbahnhof	NULL	verspätet	NULL	NULL	NULL
24	NULL	NULL	NULL	NULL	NULL	NULL

24 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT Status, COUNT(*) AS Anzahl_Lieferungen
    → FROM Lieferung
    → GROUP BY Status
    → ORDER BY Anzahl_Lieferungen DESC;
```

Status	Anzahl_Lieferungen
verspätet	7
NULL	5
geplant	4
in Bearbeitung	3
unterwegs	3
geliefert	2

6 rows in set (0,001 sec)

Durchschnittliche Versandkosten pro Transporttyp

```
SELECT Transporttyp, AVG(Versandkosten) AS Durchschnittskosten
FROM Lieferung
WHERE Versandkosten IS NOT NULL
GROUP BY Transporttyp
HAVING AVG(Versandkosten) > 15
ORDER BY Durchschnittskosten DESC;
```

Durch WHERE Versandkosten IS NOT NULL werden NULL-Werte ausgeschlossen, da AVG() sonst falsche Ergebnisse liefern kann. Mit GROUP BY Transporttyp werden Lieferungen nach Transportmittel (LKW, Drohne, etc.) gruppiert. Durch AVG(Versandkosten) wird der Durchschnitt der Versandkosten für jede Gruppe berechnet. Anhand von HAVING AVG(Versandkosten) > 15 werden nur Transporttypen mit durchschnittlichen Versandkosten von über 15 gefiltert. Durch ORDER BY Durchschnittskosten DESC werden zuerst die teuersten Transportarten gezeigt.

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
1	Berlin, Alexanderplatz	2044-06-15	unterwegs	NULL	NULL	NULL
2	Hamburg, Hafenstraße	2044-06-16	geliefert	NULL	NULL	NULL
3	München, Marienplatz	2044-06-17	verspätet	NULL	NULL	NULL
4	Frankfurt, Zeilstraße	2044-06-18	geplant	NULL	NULL	NULL
5	Unbekannt	2044-06-20	in Bearbeitung	NULL	NULL	NULL
6	Unbekannt	2044-06-21	Verspätet	NULL	NULL	NULL
7	Unbekannt	2044-06-22	Verspätet	NULL	NULL	NULL
8	Unbekannt	2044-06-23	Verspätet	NULL	NULL	NULL
9	Köln, Domplatz	2044-06-25	geplant	2	Drohne	19.99
10	NULL	2044-06-27	unterwegs	1	Elektro-LKW	29.99
11	Hamburg, Reeperbahn	NULL	verspätet	3	Hybrid-LKW	24.50
12	München, Stachus	2044-07-01	geliefert	2	Elektro-Van	15.75
13	NULL	NULL	in Bearbeitung	1	Drohne	12.99
14	Düsseldorf, Altstadt	2044-07-05	NULL	2	Wasserstoff-LKW	22.30
15	Stuttgart, Schlossplatz	2044-06-29	unterwegs	1	Elektro-Scooter	9.99
16	Frankfurt, Mainufer	NULL	NULL	3	Drohne	14.50
17	NULL	2044-07-03	geplant	2	Elektro-LKW	17.89
18	Berlin, Kudamm	NULL	in Bearbeitung	1	Hybrid-Drohne	11.99
19	NULL	NULL	NULL	3	Autonomer LKW	31.50
20	Nürnberg, Zentrum	2044-07-10	verspätet	1	Drohne	10.99
21	Bremen, Innenstadt	2044-07-12	NULL	NULL	NULL	NULL
22	NULL	2044-07-13	geplant	NULL	NULL	NULL
23	Hannover, Hauptbahnhof	NULL	verspätet	NULL	NULL	NULL
24	NULL	NULL	NULL	NULL	NULL	NULL

24 rows in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT Transporttyp, AVG(Versandkosten) AS Durchschnittskosten

```

→ FROM Lieferung
→ WHERE Versandkosten IS NOT NULL
→ GROUP BY Transporttyp
→ HAVING AVG(Versandkosten) > 15
→ ORDER BY Durchschnittskosten DESC;
```

Transporttyp	Durchschnittskosten
Autonomer LKW	31.500000
Hybrid-LKW	24.500000
Elektro-LKW	23.940000
Wasserstoff-LKW	22.300000
Elektro-Van	15.750000

5 rows in set (0,019 sec)

Anzahl der Fahrzeuge pro Typ, sortiert nach Kapazität

Der Ablauf für diese und andere Beispiele ist ähnlich. Deshalb lasse ich die Erklärungen weg.

```

SELECT Typ, COUNT(*) AS Fahrzeuganzahl, SUM(Kapazität) AS Gesamtkapazität
FROM Fahrzeug
WHERE Typ IS NOT NULL
GROUP BY Typ
HAVING COUNT(*) > 1
ORDER BY Gesamtkapazität DESC;
```

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Fahrzeug_ID | Typ | Kapazität | Batteriestatus | Lieferung_Lieferungsnummer | Wartungsstatus | Ladezeit | CO2_Einsparung |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Elektro-LKW | 5000 | 85 | 1 | NULL | NULL | NULL |
| 3 | Elektro-Van | 1100 | 75 | 3 | NULL | NULL | NULL |
| 5 | Autonomer LKW | 7000 | 100 | 3 | NULL | NULL | NULL |
| 6 | Wasserstoff-LKW | 6500 | 100 | 3 | NULL | NULL | NULL |
| 9 | Elektro-LKW | 7000 | 90 | NULL | NULL | NULL | NULL |
| 10 | NULL | 1000 | 100 | 1 | NULL | NULL | NULL |
| 11 | Hybrid-LKW | NULL | 80 | 3 | NULL | NULL | NULL |
| 12 | Drohne | 50 | NULL | 4 | NULL | NULL | NULL |
| 13 | Elektro-Van | 2000 | 60 | NULL | NULL | NULL | NULL |
| 14 | NULL |
| 15 | Autonomer LKW | 7500 | 100 | 2 | NULL | NULL | NULL |
| 16 | Wasserstoff-LKW | NULL | NULL | 5 | NULL | NULL | NULL |
| 17 | NULL | 8000 | 75 | NULL | NULL | NULL | NULL |
| 18 | Hybrid-Drohne | 85 | 95 | NULL | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
14 rows in set (0,001 sec)
```

```
MariaDB [hosakbari_db]> SELECT Typ, COUNT(*) AS Fahrzeuganzahl, SUM(Kapazität) AS Gesamtkapazität
→ FROM Fahrzeug
→ WHERE Typ IS NOT NULL
→ GROUP BY Typ
→ HAVING COUNT(*) > 1
→ ORDER BY Gesamtkapazität DESC;
```

Typ	Fahrzeuganzahl	Gesamtkapazität
Autonomer LKW	2	14500
Elektro-LKW	2	12000
Wasserstoff-LKW	2	6500
Elektro-Van	2	3100

4 rows in set (0,002 sec)

Lagerhäuser mit hoher Kapazität und Anzahl der Artikel

Dito für dieses Beispiel:

```
SELECT Lagerhaus_Lager_ID, COUNT(*) AS Artikelanzahl, SUM(Gewicht) AS Gesamtgewicht
FROM Artikel
WHERE Lagerhaus_Lager_ID IS NOT NULL
GROUP BY Lagerhaus_Lager_ID
HAVING SUM(Gewicht) > 100
ORDER BY Gesamtgewicht DESC;
```

```
MariaDB [hosakbari_db]> SELECT * FROM Artikel;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Artikelnummer | Bezeichnung | Gewicht | Lagerbestand | Lagerhaus_Lager_ID | Produktionsland | Garantiezeit | Recyclingklasse |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 101 | Smartphone-Modell X | 0.24 | 1450 | 1 | NULL | NULL | NULL |
| 102 | Laptop-Modell Y | 1.8 | 450 | 2 | NULL | NULL | NULL |
| 103 | Tablet-Modell Z | 0.96 | 750 | 3 | NULL | NULL | NULL |
| 104 | Kühlschrank Eco-Kühl | 50 | 70 | 4 | NULL | NULL | NULL |
| 105 | Waschmaschine SuperClean | 75 | 50 | 4 | NULL | NULL | NULL |
| 106 | Drucker Modell T100 | 5 | 100 | 1 | NULL | NULL | NULL |
| 107 | 3D-Drucker HighTech | 10 | 150 | 1 | NULL | NULL | NULL |
| 108 | Neues Produkt | 0.6 | 250 | 1 | NULL | NULL | NULL |
| 109 | Smartwatch Pro | 0.36 | 450 | 1 | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
9 rows in set (0,001 sec)
```

```
MariaDB [hosakbari_db]> SELECT Lagerhaus_Lager_ID, COUNT(*) AS Artikelanzahl, SUM(Gewicht) AS Gesamtgewicht
→ FROM Artikel
→ WHERE Lagerhaus_Lager_ID IS NOT NULL
→ GROUP BY Lagerhaus_Lager_ID
→ HAVING SUM(Gewicht) > 100
→ ORDER BY Gesamtgewicht DESC;
```

Lagerhaus_Lager_ID	Artikelanzahl	Gesamtgewicht
4	2	125

1 row in set (0,021 sec)

Durchschnittlicher Ladestatus von Fahrzeugen pro Fahrzeugtyp

Dito hierfür:

```
SELECT Typ, AVG(Batteriestatus) AS Durchschnitts_Ladestatus
FROM Fahrzeug
WHERE Batteriestatus IS NOT NULL
GROUP BY Typ
HAVING AVG(Batteriestatus) > 50
ORDER BY Durchschnitts_Ladestatus DESC;
```

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
```

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer	Wartungsstatus	Ladezeit	CO2_Einsparung
1	Elektro-LKW	5000	85		1	NULL	NULL
3	Elektro-Van	1100	75		3	NULL	NULL
5	Autonomer LKW	7000	100		3	NULL	NULL
6	Wasserstoff-LKW	6500	100		3	NULL	NULL
9	Elektro-LKW	7000	90		NULL	NULL	NULL
10	NULL	1000	100		1	NULL	NULL
11	Hybrid-LKW	NULL	80		3	NULL	NULL
12	Drohne	50	NULL		4	NULL	NULL
13	Elektro-Van	2000	60		NULL	NULL	NULL
14	NULL	NULL	NULL		NULL	NULL	NULL
15	Autonomer LKW	7500	100		2	NULL	NULL
16	Wasserstoff-LKW	NULL	NULL		5	NULL	NULL
17	NULL	8000	75		NULL	NULL	NULL
18	Hybrid-Drohne	85	95		NULL	NULL	NULL

14 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT Typ, AVG(Batteriestatus) AS Durchschnitts_Ladestatus
```

```
→ FROM Fahrzeug  
→ WHERE Batteriestatus IS NOT NULL  
→ GROUP BY Typ  
→ HAVING AVG(Batteriestatus) > 50  
→ ORDER BY Durchschnitts_Ladestatus DESC;
```

Typ	Durchschnitts_Ladestatus
Wasserstoff-LKW	100.0000
Autonomer LKW	100.0000
Hybrid-Drohne	95.0000
Elektro-LKW	87.5000
NULL	87.5000
Hybrid-LKW	80.0000
Elektro-Van	67.5000

7 rows in set (0,001 sec)

Problematik „Logistik in 20 Jahren“; Führen Sie folgende SQL-Operationen (2 Beispiele pro Operation) mit Ihren Testdaten aus: Mengen-Operationen (OR), Mengen-Operationen – AND, Der natürliche Verbund (Join), INNER JOIN, Kartesisches Produkt (CROSS – JOIN), AS, Linker äußerer (Left Outer) Join, Rechter äußerer (Right Outer) Join, Right Join

Wie in der Aufgabenstellung beschrieben, müssen die obigen Befehle auf die Testdaten angewendet werden.

Mengen-Operationen (OR)

Bei der OR-Bedingung werden Datensätze zurückgegeben, wenn mindestens eine der angegebenen Bedingungen zutrifft. Beispiele folgen:

Fahrzeuge mit niedriger Kapazität ORDER leerem Batteriestatus

`SELECT * FROM Fahrzeug`

`WHERE Kapazität < 2000 OR Batteriestatus IS NULL;`

Bei der obigen Operation werden Fahrzeuge mit niedriger Kapazität (weniger als 2000) oder Fahrzeuge mit unbekanntem Batteriestatus (NULL) gezeigt.

MariaDB [hosakbari_db]> `SELECT * FROM Fahrzeug;`

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer	Wartungsstatus	Ladezeit	CO2_Einsparung
1	Elektro-LKW	5000	85		1	NULL	NULL
3	Elektro-Van	1100	75		3	NULL	NULL
5	Autonomer LKW	7000	100		3	NULL	NULL
6	Wasserstoff-LKW	6500	100		3	NULL	NULL
9	Elektro-LKW	7000	90		NULL	NULL	NULL
10	NULL	1000	100		1	NULL	NULL
11	Hybrid-LKW	NULL	80		3	NULL	NULL
12	Drohne	50	NULL		4	NULL	NULL
13	Elektro-Van	2000	60		NULL	NULL	NULL
14	NULL	NULL	NULL		NULL	NULL	NULL
15	Autonomer LKW	7500	100		2	NULL	NULL
16	Wasserstoff-LKW	NULL	NULL		5	NULL	NULL
17	NULL	8000	75		NULL	NULL	NULL
18	Hybrid-Drohne	85	95		NULL	NULL	NULL

14 rows in set (0,001 sec)

MariaDB [hosakbari_db]> `SELECT * FROM Fahrzeug`

`→ WHERE Kapazität < 2000 OR Batteriestatus IS NULL;`

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer	Wartungsstatus	Ladezeit	CO2_Einsparung
3	Elektro-Van	1100	75		3	NULL	NULL
10	NULL	1000	100		1	NULL	NULL
12	Drohne	50	NULL		4	NULL	NULL
14	NULL	NULL	NULL		NULL	NULL	NULL
16	Wasserstoff-LKW	NULL	NULL		5	NULL	NULL
18	Hybrid-Drohne	85	95		NULL	NULL	NULL

6 rows in set (0,001 sec)

Lieferungen nach Berlin ODER Lieferungen mit einem hohen Versandkostenbetrag

`SELECT * FROM Lieferung`

`WHERE Zielort LIKE '%Berlin%' OR Versandkosten > 25;`

Bei der obigen Operation werden Lieferungen nach Berlin gezeigt, oder Lieferungen mit hohen Versandkosten.

MariaDB [hosakbari_db]> SELECT * FROM Lieferung;

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
1	Berlin, Alexanderplatz	2044-06-15	unterwegs	NULL	NULL	NULL
2	Hamburg, Hafenstraße	2044-06-16	geliefert	NULL	NULL	NULL
3	München, Marienplatz	2044-06-17	verspätet	NULL	NULL	NULL
4	Frankfurt, Zeilstraße	2044-06-18	geplant	NULL	NULL	NULL
5	Unbekannt	2044-06-20	in Bearbeitung	NULL	NULL	NULL
6	Unbekannt	2044-06-21	Verspätet	NULL	NULL	NULL
7	Unbekannt	2044-06-22	Verspätet	NULL	NULL	NULL
8	Unbekannt	2044-06-23	Verspätet	NULL	NULL	NULL
9	Köln, Domplatz	2044-06-25	geplant	2	Drohne	19.99
10	NUL	2044-06-27	unterwegs	1	Elektro-LKW	29.99
11	Hamburg, Reeperbahn	NULL	verspätet	3	Hybrid-LKW	24.50
12	München, Stachus	2044-07-01	geliefert	2	Elektro-Van	15.75
13	NUL	NULL	in Bearbeitung	1	Drohne	12.99
14	Düsseldorf, Altstadt	2044-07-05	NULL	2	Wasserstoff-LKW	22.30
15	Stuttgart, Schlossplatz	2044-06-29	unterwegs	1	Elektro-Scooter	9.99
16	Frankfurt, Mainufer	NULL	NULL	3	Drohne	14.50
17	NUL	2044-07-03	geplant	2	Elektro-LKW	17.89
18	Berlin, Kudamm	NULL	in Bearbeitung	1	Hybrid-Drohne	11.99
19	NUL	NULL	NULL	3	Autonomer LKW	31.50
20	Nürnberg, Zentrum	2044-07-10	verspätet	1	Drohne	10.99
21	Bremen, Innenstadt	2044-07-12	NULL	NULL	NULL	NULL
22	NUL	2044-07-13	geplant	NULL	NULL	NULL
23	Hannover, Hauptbahnhof	NULL	verspätet	NULL	NULL	NULL
24	NUL	NULL	NULL	NULL	NULL	NULL

24 rows in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT * FROM Lieferung
→ WHERE Zielort LIKE '%Berlin%' OR Versandkosten > 25;

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
1	Berlin, Alexanderplatz	2044-06-15	unterwegs	NULL	NULL	NULL
10	NUL	2044-06-27	unterwegs	1	Elektro-LKW	29.99
18	Berlin, Kudamm	NULL	in Bearbeitung	1	Hybrid-Drohne	11.99
19	NUL	NULL	NULL	3	Autonomer LKW	31.50

4 rows in set (0,001 sec)

Artikel mit niedrigem Lagerbestand ODER ohne zugewiesenes Lagerhaus

SELECT * FROM Artikel

WHERE Lagerbestand < 100 OR Lagerhaus_Lager_ID IS NULL;

Bei der obigen Operation werden Artikel aufgelistet, die entweder fast ausverkauft sind oder keinem Lagerhaus zugeordnet wurden.

MariaDB [hosakbari_db]> SELECT * FROM Artikel;

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID	Produktionsland	Garantiezeit	Recyclingklasse
101	Smartphone-Modell X	0.24	1450	1	NULL	NULL	NULL
102	Laptop-Modell Y	1.8	450	2	NULL	NULL	NULL
103	Tablet-Modell Z	0.96	750	3	NULL	NULL	NULL
104	Kühlschrank Eco-Kühl	50	70	4	NULL	NULL	NULL
105	Waschmaschine SuperClean	75	50	4	NULL	NULL	NULL
106	Drucker Modell T100	5	100	1	NULL	NULL	NULL
107	3D-Drucker HighTech	10	150	1	NULL	NULL	NULL
108	Neues Produkt	0.6	250	1	NULL	NULL	NULL
109	Smartwatch Pro	0.36	450	1	NULL	NULL	NULL

9 rows in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT * FROM Artikel WHERE Lagerbestand < 100 OR Lagerhaus_Lager_ID IS NULL;

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID	Produktionsland	Garantiezeit	Recyclingklasse
104	Kühlschrank Eco-Kühl	50	70	4	NULL	NULL	NULL
105	Waschmaschine SuperClean	75	50	4	NULL	NULL	NULL

2 rows in set (0,001 sec)

Lagerhäuser mit erneuerbarer Energiequelle ODER Kapazität über 50.000

SELECT * FROM Lagerhaus

WHERE Energiequelle = 'Erneuerbar' OR Kapazität > 50000;

Die obige Operation zeigt alle umweltfreundlichen Lagerhäuser, oder Lager mit hoher Kapazität.

Lager_ID	Standort	Kapazität	Energiequelle	Sicherheitsstufe	Kühlkapazität	Drohnenlandeplatz
1	Berlin, Lagerstraße 12	50000	Solar	NULL	NULL	NULL
2	Hamburg, Hafenlager 23	75000	Wind	NULL	NULL	NULL
3	München, Südallee 45	35000	Solar	NULL	NULL	NULL
4	Frankfurt, Westhafen 8	100000	Solar	NULL	NULL	NULL
5	Leipzig, Zentrallager	45000	Erneuerbar	NULL	NULL	NULL
6	Stuttgart, Hauptlager 1	55000	Erneuerbar	NULL	NULL	NULL
7	Düsseldorf, Logistikzentrum	50000	Erneuerbar	NULL	NULL	NULL
8	Dresden, Verteilzentrum	60000	Erneuerbar	NULL	NULL	NULL

8 rows in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus
→ WHERE Energiequelle = 'Erneuerbar' OR Kapazität > 50000;

Lager_ID	Standort	Kapazität	Energiequelle	Sicherheitsstufe	Kühlkapazität	Drohnenlandeplatz
2	Hamburg, Hafenlager 23	75000	Wind	NULL	NULL	NULL
4	Frankfurt, Westhafen 8	100000	Solar	NULL	NULL	NULL
5	Leipzig, Zentrallager	45000	Erneuerbar	NULL	NULL	NULL
6	Stuttgart, Hauptlager 1	55000	Erneuerbar	NULL	NULL	NULL
7	Düsseldorf, Logistikzentrum	50000	Erneuerbar	NULL	NULL	NULL
8	Dresden, Verteilzentrum	60000	Erneuerbar	NULL	NULL	NULL

6 rows in set (0,001 sec)

Mengen-Operationen (AND)

Die AND-Bedingung sorgt für Datensätze die zurückgegeben werden, wenn gleichwohl alle Bedingungen zutreffen. Beispiele folgen:

Fahrzeuge mit hoher Kapazität UND vollem Batteriestatus

SELECT * FROM Fahrzeug

WHERE Kapazität > 6000 AND Batteriestatus = 100;

Die Operation oben zeigt Fahrzeuge mit hoher Ladekapazität, sowie die die vollständig geladen sind.

MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer	Wartungsstatus	Ladezeit	CO2_Einsparung
1	Elektro-LKW	5000	85	1	NULL	NULL	NULL
3	Elektro-Van	11000	75	3	NULL	NULL	NULL
5	Autonomer LKW	7000	100	3	NULL	NULL	NULL
6	Wasserstoff-LKW	6500	100	3	NULL	NULL	NULL
9	Elektro-LKW	7000	90	NULL	NULL	NULL	NULL
10	NULL	10000	100	1	NULL	NULL	NULL
11	Hybrid-LKW	NULL	80	3	NULL	NULL	NULL
12	Drohne	50	NULL	4	NULL	NULL	NULL
13	Elektro-Van	2000	60	NULL	NULL	NULL	NULL
14	NULL	NULL	NULL	NULL	NULL	NULL	NULL
15	Autonomer LKW	7500	100	2	NULL	NULL	NULL
16	Wasserstoff-LKW	NULL	NULL	5	NULL	NULL	NULL
17	NULL	8000	75	NULL	NULL	NULL	NULL
18	Hybrid-Drohne	85	95	NULL	NULL	NULL	NULL

14 rows in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug
→ WHERE Kapazität > 6000 AND Batteriestatus = 100;

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer	Wartungsstatus	Ladezeit	CO2_Einsparung
5	Autonomer LKW	7000	100	3	NULL	NULL	NULL
6	Wasserstoff-LKW	6500	100	3	NULL	NULL	NULL
15	Autonomer LKW	7500	100	2	NULL	NULL	NULL

3 rows in set (0,001 sec)

Lieferungen mit geringem Versandkostenbetrag UND geplantem Status

SELECT * FROM Lieferung

WHERE Versandkosten < 20 AND Status = 'geplant';

Die obige Operation listet kostengünstige und geplante Lieferungen auf.

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
1	Berlin, Alexanderplatz	2044-06-15	unterwegs	NULL	NULL	NULL
2	Hamburg, Hafenstraße	2044-06-16	geliefert	NULL	NULL	NULL
3	München, Marienplatz	2044-06-17	verspätet	NULL	NULL	NULL
4	Frankfurt, Zeilstraße	2044-06-18	geplant	NULL	NULL	NULL
5	Unbekannt	2044-06-20	in Bearbeitung	NULL	NULL	NULL
6	Unbekannt	2044-06-21	Verspätet	NULL	NULL	NULL
7	Unbekannt	2044-06-22	Verspätet	NULL	NULL	NULL
8	Unbekannt	2044-06-23	Verspätet	NULL	NULL	NULL
9	Köln, Domplatz	2044-06-25	geplant	2	Drohne	19.99
10	NUL	2044-06-27	unterwegs	1	Elektro-LKW	29.99
11	Hamburg, Reeperbahn	NULL	verspätet	3	Hybrid-LKW	24.50
12	München, Stachus	2044-07-01	geliefert	2	Elektro-Van	15.75
13	NUL	NULL	in Bearbeitung	1	Drohne	12.99
14	Düsseldorf, Altstadt	2044-07-05	NULL	2	Wasserstoff-LKW	22.30
15	Stuttgart, Schlossplatz	2044-06-29	unterwegs	1	Elektro-Scooter	9.99
16	Frankfurt, Mainufer	NULL	NULL	3	Drohne	14.50
17	NUL	2044-07-03	geplant	2	Elektro-LKW	17.89
18	Berlin, Kudamm	NULL	in Bearbeitung	1	Hybrid-Drohne	11.99
19	NUL	NULL	NULL	3	Autonomer LKW	31.50
20	Nürnberg, Zentrum	2044-07-10	verspätet	1	Drohne	18.99
21	Bremen, Innenstadt	2044-07-12	NULL	NULL	NULL	NULL
22	NUL	2044-07-13	geplant	NULL	NULL	NULL
23	Hannover, Hauptbahnhof	NULL	verspätet	NULL	NULL	NULL
24	NUL	NULL	NULL	NULL	NULL	NULL

24 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung
    → WHERE Versandkosten < 20 AND Status = 'geplant';
```

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
9	Köln, Domplatz	2044-06-25	geplant	2	Drohne	19.99
17	NUL	2044-07-03	geplant	2	Elektro-LKW	17.89

2 rows in set (0,001 sec)

Artikel mit hohem Lagerbestand UND bekanntem Produktionsland

SELECT * FROM Artikel

WHERE Lagerbestand > 1000 AND Produktionsland IS NOT NULL;

Die obige Operation zeigt gut verfügbare Artikel, sowie zugleich die die einen bekannten Herkunftsland haben.

```
MariaDB [hosakbari_db]> SELECT * FROM Artikel;
```

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID	Produktionsland	Garantiezeit	Recyclingklasse
101	Smartphone-Modell X	0.24	1450	1	NULL	NULL	NULL
102	Laptop-Modell Y	1.8	450	2	NULL	NULL	NULL
103	Tablet-Modell Z	0.96	750	3	NULL	NULL	NULL
104	Kühlschrank Eco-Kühl	50	70	4	NULL	NULL	NULL
105	Waschmaschine SuperClean	75	50	4	NULL	NULL	NULL
106	Drucker Modell T100	5	100	1	NULL	NULL	NULL
107	3D-Drucker HighTech	10	150	1	NULL	NULL	NULL
108	Neues Produkt	0.6	250	1	NULL	NULL	NULL
109	Smartwatch Pro	0.36	450	1	NULL	NULL	NULL

9 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Artikel
    → WHERE Lagerbestand > 1000 AND Produktionsland IS NOT NULL;
Empty set (0,001 sec)
```

Hierzu gibt es keine Treffer.

Lagerhäuser mit erneuerbarer Energiequelle UND Kühlkapazität

SELECT * FROM Lagerhaus

WHERE Energiequelle = 'Erneuerbar' AND Kühlkapazität IS NOT NULL;

Bei der obigen Operation werden umweltfreundliche Lager mit Kühlung gezeigt.

MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;

Lager_ID	Standort	Kapazität	Energiequelle	Sicherheitsstufe	Kühlkapazität	Drohnenlandeplatz
1	Berlin, Lagerstraße 12	50000	Solar	NULL	NULL	NULL
2	Hamburg, Hafenlager 23	75000	Wind	NULL	NULL	NULL
3	München, Südallee 45	35000	Solar	NULL	NULL	NULL
4	Frankfurt, Westhafen 8	100000	Solar	NULL	NULL	NULL
5	Leipzig, Zentrallager	45000	Erneuerbar	NULL	NULL	NULL
6	Stuttgart, Hauptlager 1	55000	Erneuerbar	NULL	NULL	NULL
7	Düsseldorf, Logistikzentrum	50000	Erneuerbar	NULL	NULL	NULL
8	Dresden, Verteilzentrum	60000	Erneuerbar	NULL	NULL	NULL

8 rows in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus
→ WHERE Energiequelle = 'Erneuerbar' AND Kühlkapazität IS NOT NULL;
Empty set (0,001 sec)

NATURAL JOIN (Natürlicher Verbund)

Ein NATURAL JOIN verbindet zwei Tabellen automatisch über gleichnamige Spalten. Beispiele folgen:

Fahrzeuge mit zugewiesener Lieferung

SELECT * FROM Fahrzeug NATURAL JOIN Lieferung;

Die Operation oben verknüpft die Tabellen Fahrzeug und Lieferung automatisch über die gemeinsame Spalte Lieferung_Lieferungsnummer.

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer	Wartungsstatus	Ladezeit	CO2_Einsparung	Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
1	Elektro-LKW	50000	85	1	NULL	NULL	NULL	1	Berlin, Alexanderplatz	2804-06-15	unterwegs	NULL	NULL	NULL
2	Autonomer LKW	11000	75	1	NULL	NULL	NULL	1	Berlin, Alexanderplatz	2804-06-15	unterwegs	NULL	NULL	NULL
3	Hybrid-LKW	70000	100	3	NULL	NULL	NULL	1	Berlin, Alexanderplatz	2804-06-15	unterwegs	NULL	NULL	NULL
4	Wasserstoff-LKW	65000	100	3	NULL	NULL	NULL	1	Berlin, Alexanderplatz	2804-06-15	unterwegs	NULL	NULL	NULL
5	NULL	10000	90	NULL	NULL	NULL	NULL	1	Berlin, Alexanderplatz	2804-06-15	unterwegs	NULL	NULL	NULL
6	Hybrid-LKW	10000	100	1	NULL	NULL	NULL	1	Berlin, Alexanderplatz	2804-06-15	unterwegs	NULL	NULL	NULL
7	Autonomer LKW	NULL	80	3	NULL	NULL	NULL	1	Berlin, Alexanderplatz	2804-06-15	unterwegs	NULL	NULL	NULL
8	Drone	NULL	100	1	NULL	NULL	NULL	1	Berlin, Alexanderplatz	2804-06-15	unterwegs	NULL	NULL	NULL
9	Elektro-Van	26000	60	NULL	NULL	NULL	NULL	1	Berlin, Alexanderplatz	2804-06-15	unterwegs	NULL	NULL	NULL
10	NULL	NULL	NULL	1	NULL	NULL	NULL	1	Berlin, Alexanderplatz	2804-06-15	unterwegs	NULL	NULL	NULL
11	Hybrid-Van	26000	60	NULL	NULL	NULL	NULL	1	Berlin, Alexanderplatz	2804-06-15	unterwegs	NULL	NULL	NULL
12	Autonomer LKW	72000	100	1	NULL	NULL	NULL	1	Berlin, Alexanderplatz	2804-06-15	unterwegs	NULL	NULL	NULL
13	Wasserstoff-LKW	65000	100	1	NULL	NULL	NULL	1	Berlin, Alexanderplatz	2804-06-15	unterwegs	NULL	NULL	NULL
14	NULL	80000	75	NULL	NULL	NULL	NULL	1	Berlin, Alexanderplatz	2804-06-15	unterwegs	NULL	NULL	NULL
15	Hybrid-Drohne	10000	95	NULL	NULL	NULL	NULL	1	Berlin, Alexanderplatz	2804-06-15	unterwegs	NULL	NULL	NULL
16	Elektro-Van	50000	95	3	NULL	NULL	NULL	2	Hamburg, Hafenstraße	2804-06-16	geliefert	NULL	NULL	NULL
17	Autonomer LKW	11000	75	3	NULL	NULL	NULL	2	Hamburg, Hafenstraße	2804-06-16	geliefert	NULL	NULL	NULL
18	Hybrid-Drohne	70000	100	3	NULL	NULL	NULL	2	Hamburg, Hafenstraße	2804-06-16	geliefert	NULL	NULL	NULL
19	Wasserstoff-Van	65000	100	3	NULL	NULL	NULL	2	Hamburg, Hafenstraße	2804-06-16	geliefert	NULL	NULL	NULL
20	Autonomer LKW	70000	90	NULL	NULL	NULL	NULL	2	Hamburg, Hafenstraße	2804-06-16	geliefert	NULL	NULL	NULL
21	Hybrid-LKW	10000	80	3	NULL	NULL	NULL	2	Hamburg, Hafenstraße	2804-06-16	geliefert	NULL	NULL	NULL
22	Autonomer LKW	75000	100	2	NULL	NULL	NULL	2	Hamburg, Hafenstraße	2804-06-16	geliefert	NULL	NULL	NULL
23	Wasserstoff-LKW	65000	100	5	NULL	NULL	NULL	2	Hamburg, Hafenstraße	2804-06-16	geliefert	NULL	NULL	NULL
24	NULL	80000	95	NULL	NULL	NULL	NULL	2	Hamburg, Hafenstraße	2804-06-16	geliefert	NULL	NULL	NULL
25	Hybrid-Drohne	10000	95	NULL	NULL	NULL	NULL	2	Hamburg, Hafenstraße	2804-06-16	geliefert	NULL	NULL	NULL
26	Elektro-Van	26000	60	NULL	NULL	NULL	NULL	3	München, Marienplatz	2804-06-17	verspätet	NULL	NULL	NULL
27	Autonomer LKW	75000	100	2	NULL	NULL	NULL	3	München, Marienplatz	2804-06-17	verspätet	NULL	NULL	NULL
28	Hybrid-Drohne	70000	100	3	NULL	NULL	NULL	3	München, Marienplatz	2804-06-17	verspätet	NULL	NULL	NULL
29	Wasserstoff-Van	65000	100	3	NULL	NULL	NULL	3	München, Marienplatz	2804-06-17	verspätet	NULL	NULL	NULL
30	Autonomer LKW	70000	90	NULL	NULL	NULL	NULL	3	München, Marienplatz	2804-06-17	verspätet	NULL	NULL	NULL
31	Hybrid-LKW	10000	80	3	NULL	NULL	NULL	3	München, Marienplatz	2804-06-17	verspätet	NULL	NULL	NULL
32	Autonomer LKW	75000	100	2	NULL	NULL	NULL	3	München, Marienplatz	2804-06-17	verspätet	NULL	NULL	NULL
33	Wasserstoff-LKW	65000	100	5	NULL	NULL	NULL	3	München, Marienplatz	2804-06-17	verspätet	NULL	NULL	NULL
34	NULL	80000	95	NULL	NULL	NULL	NULL	3	München, Marienplatz	2804-06-17	verspätet	NULL	NULL	NULL
35	Hybrid-Drohne	10000	95	NULL	NULL	NULL	NULL	3	München, Marienplatz	2804-06-17	verspätet	NULL	NULL	NULL
36	Elektro-Van	26000	60	NULL	NULL	NULL	NULL	4	Frankfurt, Zeilstraße	2804-06-18	geplant	NULL	NULL	NULL
37	Autonomer LKW	11000	75	1	NULL	NULL	NULL	4	Frankfurt, Zeilstraße	2804-06-18	geplant	NULL	NULL	NULL
38	Hybrid-Drohne	70000	100	3	NULL	NULL	NULL	4	Frankfurt, Zeilstraße	2804-06-18	geplant	NULL	NULL	NULL
39	Wasserstoff-Van	65000	100	3	NULL	NULL	NULL	4	Frankfurt, Zeilstraße	2804-06-18	geplant	NULL	NULL	NULL
40	Autonomer LKW	70000	90	NULL	NULL	NULL	NULL	4	Frankfurt, Zeilstraße	2804-06-18	geplant	NULL	NULL	NULL
41	Hybrid-LKW	10000	80	3	NULL	NULL	NULL	4	Frankfurt, Zeilstraße	2804-06-18	geplant	NULL	NULL	NULL
42	Autonomer LKW	75000	100	2	NULL	NULL	NULL	4	Frankfurt, Zeilstraße	2804-06-18	geplant	NULL	NULL	NULL
43	Wasserstoff-LKW	65000	100	5	NULL	NULL	NULL	4	Frankfurt, Zeilstraße	2804-06-18	geplant	NULL	NULL	NULL
44	NULL	80000	95	NULL	NULL	NULL	NULL	4	Frankfurt, Zeilstraße	2804-06-18	geplant	NULL	NULL	NULL
45	Hybrid-Drohne	10000	95	NULL	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
46	Elektro-Van	26000	60	NULL	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
47	Autonomer LKW	50000	85	3	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
48	Hybrid-LKW	65000	100	3	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
49	Autonomer LKW	70000	90	NULL	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
50	Hybrid-Drohne	70000	100	1	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
51	Elektro-Van	26000	60	3	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
52	Autonomer LKW	75000	100	3	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
53	Hybrid-Drohne	70000	100	3	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
54	Elektro-Van	26000	60	3	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
55	Autonomer LKW	75000	100	2	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
56	Hybrid-LKW	65000	100	5	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
57	Autonomer LKW	70000	90	NULL	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
58	Hybrid-Drohne	70000	100	1	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
59	Elektro-Van	26000	60	3	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
60	Autonomer LKW	75000	100	3	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
61	Hybrid-Drohne	70000	100	3	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
62	Elektro-Van	26000	60	3	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
63	Autonomer LKW	75000	100	2	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
64	Hybrid-LKW	65000	100	5	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
65	Autonomer LKW	70000	90	NULL	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
66	Hybrid-Drohne	70000	100	1	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
67	Elektro-Van	26000	60	3	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
68	Autonomer LKW	75000	100	3	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
69	Hybrid-Drohne	70000	100	3	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
70	Elektro-Van	26000	60	3	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
71	Autonomer LKW	75000	100	2	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
72	Hybrid-LKW	65000	100	5	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
73	Autonomer LKW	70000	90	NULL	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
74	Hybrid-Drohne	70000	100	1	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
75	Elektro-Van	26000	60	3	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
76	Autonomer LKW	75000	100	3	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
77	Hybrid-Drohne	70000	100	3	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
78	Elektro-Van	26000	60	3	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
79	Autonomer LKW	75000	100	2	NULL	NULL	NULL	5	Unbekannt	2804-06-19	in Bearbeitung	NULL	NULL	NULL
80	Hybrid-LKW	65000	100	5	NULL	NULL	NULL	5	Unbekannt	2				

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID	Produktionsland	Garantiezeit	Recyclingklasse	Lager_ID	Standort	Kapazität	Energiequelle	Sicherheitsstufe	Kühlkapazität	DrehenLandeplatz
101	Smartphone-Modell X	0.24	1450	1	MALL	MALL		1	Berlin, Lagerstraße 12	56000	Solar	MALL	MALL	MULL
101	Smartphone-Modell X	0.24	1450	1	MALL	MALL		2	Hamburg, Hafenlager 23	75000	Wind	MALL	MALL	MULL
101	Smartphone-Modell X	0.24	1450	1	MALL	MALL		3	München, Südalles 45	35000	Solar	MALL	MALL	MULL
101	Smartphone-Modell X	0.24	1450	1	MALL	MALL		4	Düsseldorf, Logistikzentrum	100000	Solar	MALL	MALL	MULL
101	Smartphone-Modell X	0.24	1450	1	MALL	MALL		5	Leipzig, Zentrallager	45000	Erneuerbar	MALL	MALL	MULL
101	Smartphone-Modell X	0.24	1450	1	MALL	MALL		6	Stuttgart, Hauptlager 1	55000	Erneuerbar	MALL	MALL	MULL
101	Smartphone-Modell X	0.24	1450	1	MALL	MALL		7	Düsseldorf, Logistikzentrum	50000	Erneuerbar	MALL	MALL	MULL
101	Smartphone-Modell X	0.24	1450	1	MALL	MALL		8	Trier, Vertriebszentrum	60000	Erneuerbar	MALL	MALL	MULL
102	Laptop-Modell Y	1.8	450	2	MALL	MALL		1	Berlin, Lagerstraße 12	56000	Solar	MALL	MALL	MULL
102	Laptop-Modell Y	1.8	450	2	MALL	MALL		2	Hamburg, Hafenlager 23	75000	Wind	MALL	MALL	MULL
102	Laptop-Modell Y	1.8	450	2	MALL	MALL		3	München, Südalles 45	35000	Solar	MALL	MALL	MULL
102	Laptop-Modell Y	1.8	450	2	MALL	MALL		4	Frankfurt, Westhafen 8	100000	Solar	MALL	MALL	MULL
102	Laptop-Modell Y	1.8	450	2	MALL	MALL		5	Leipzig, Zentrallager	45000	Erneuerbar	MALL	MALL	MULL
102	Laptop-Modell Y	1.8	450	2	MALL	MALL		6	Stuttgart, Hauptlager 1	55000	Erneuerbar	MALL	MALL	MULL
102	Laptop-Modell Y	1.8	450	2	MALL	MALL		7	Düsseldorf, Logistikzentrum	50000	Erneuerbar	MALL	MALL	MULL
102	Laptop-Modell Y	1.8	450	2	MALL	MALL		8	Dresden, Vertriebszentrum	60000	Erneuerbar	MALL	MALL	MULL
103	Tablet-Modell Z	0.96	750	3	MALL	MALL		1	Berlin, Lagerstraße 12	56000	Solar	MALL	MALL	MULL
103	Tablet-Modell Z	0.96	750	3	MALL	MALL		2	Hamburg, Hafenlager 23	75000	Wind	MALL	MALL	MULL
103	Tablet-Modell Z	0.96	750	3	MALL	MALL		3	München, Südalles 45	35000	Solar	MALL	MALL	MULL
103	Tablet-Modell Z	0.96	750	3	MALL	MALL		4	Frankfurt, Westhafen 8	100000	Solar	MALL	MALL	MULL
103	Tablet-Modell Z	0.96	750	3	MALL	MALL		5	Leipzig, Zentrallager	45000	Erneuerbar	MALL	MALL	MULL
103	Tablet-Modell Z	0.96	750	3	MALL	MALL		6	Stuttgart, Hauptlager 1	55000	Erneuerbar	MALL	MALL	MULL
103	Tablet-Modell Z	0.96	750	3	MALL	MALL		7	Düsseldorf, Logistikzentrum	50000	Erneuerbar	MALL	MALL	MULL
103	Tablet-Modell Z	0.96	750	3	MALL	MALL		8	Dresden, Vertriebszentrum	60000	Erneuerbar	MALL	MALL	MULL
104	Kühlschrank Eco-Kühl	59	78	4	MALL	MALL		1	Berlin, Lagerstraße 12	56000	Solar	MALL	MALL	MULL
104	Kühlschrank Eco-Kühl	59	78	4	MALL	MALL		2	Hamburg, Hafenlager 23	75000	Wind	MALL	MALL	MULL
104	Kühlschrank Eco-Kühl	59	78	4	MALL	MALL		3	München, Südalles 45	35000	Solar	MALL	MALL	MULL
104	Kühlschrank Eco-Kühl	59	78	4	MALL	MALL		4	Frankfurt, Westhafen 8	100000	Solar	MALL	MALL	MULL
104	Kühlschrank Eco-Kühl	59	78	4	MALL	MALL		5	Leipzig, Zentrallager	45000	Erneuerbar	MALL	MALL	MULL
104	Kühlschrank Eco-Kühl	59	78	4	MALL	MALL		6	Stuttgart, Hauptlager 1	55000	Erneuerbar	MALL	MALL	MULL
104	Kühlschrank Eco-Kühl	59	78	4	MALL	MALL		7	Düsseldorf, Logistikzentrum	50000	Erneuerbar	MALL	MALL	MULL
104	Kühlschrank Eco-Kühl	59	78	4	MALL	MALL		8	Dresden, Vertriebszentrum	60000	Erneuerbar	MALL	MALL	MULL
105	Maschmaschine SuperClean	75	56	4	MALL	MALL		1	Berlin, Lagerstraße 12	56000	Solar	MALL	MALL	MULL
105	Maschmaschine SuperClean	75	56	4	MALL	MALL		2	Hamburg, Hafenlager 23	75000	Wind	MALL	MALL	MULL
105	Maschmaschine SuperClean	75	56	4	MALL	MALL		3	München, Südalles 45	35000	Solar	MALL	MALL	MULL
105	Maschmaschine SuperClean	75	56	4	MALL	MALL		4	Frankfurt, Westhafen 8	100000	Solar	MALL	MALL	MULL
105	Maschmaschine SuperClean	75	56	4	MALL	MALL		5	Leipzig, Zentrallager	45000	Erneuerbar	MALL	MALL	MULL
105	Maschmaschine SuperClean	75	56	4	MALL	MALL		6	Stuttgart, Hauptlager 1	55000	Erneuerbar	MALL	MALL	MULL
105	Maschmaschine SuperClean	75	56	4	MALL	MALL		7	Düsseldorf, Logistikzentrum	50000	Erneuerbar	MALL	MALL	MULL
105	Maschmaschine SuperClean	75	56	4	MALL	MALL		8	Dresden, Vertriebszentrum	60000	Erneuerbar	MALL	MALL	MULL
106	Drucker Modell T100	5	160	1	MALL	MALL		1	Berlin, Lagerstraße 12	56000	Solar	MALL	MALL	MULL
106	Drucker Modell T100	5	160	1	MALL	MALL		2	Hamburg, Hafenlager 23	75000	Wind	MALL	MALL	MULL
106	Drucker Modell T100	5	160	1	MALL	MALL		3	München, Südalles 45	35000	Solar	MALL	MALL	MULL
106	Drucker Modell T100	5	160	1	MALL	MALL		4	Frankfurt, Westhafen 8	100000	Solar	MALL	MALL	MULL
106	Drucker Modell T100	5	160	1	MALL	MALL		5	Leipzig, Zentrallager	45000	Erneuerbar	MALL	MALL	MULL
106	Drucker Modell T100	5	160	1	MALL	MALL		6	Stuttgart, Hauptlager 1	55000	Erneuerbar	MALL	MALL	MULL
106	Drucker Modell T100	5	160	1	MALL	MALL		7	Düsseldorf, Logistikzentrum	50000	Erneuerbar	MALL	MALL	MULL
106	Drucker Modell T100	5	160	1	MALL	MALL		8	Dresden, Vertriebszentrum	60000	Erneuerbar	MALL	MALL	MULL
107	3D-Drucker HighTech	10	150	1	MALL	MALL		1	Berlin, Lagerstraße 12	56000	Solar	MALL	MALL	MULL
107	3D-Drucker HighTech	10	150	1	MALL	MALL		2	Hamburg, Hafenlager 23	75000	Wind	MALL	MALL	MULL
107	3D-Drucker HighTech	10	150	1	MALL	MALL		3	München, Südalles 45	35000	Solar	MALL	MALL	MULL
107	3D-Drucker HighTech	10	150	1	MALL	MALL		4	Frankfurt, Westhafen 8	100000	Solar	MALL	MALL	MULL
107	3D-Drucker HighTech	10	150	1	MALL	MALL		5	Leipzig, Zentrallager	45000	Erneuerbar	MALL	MALL	MULL
107	3D-Drucker HighTech	10	150	1	MALL	MALL		6	Stuttgart, Hauptlager 1	55000	Erneuerbar	MALL	MALL	MULL
107	3D-Drucker HighTech	10	150	1	MALL	MALL		7	Düsseldorf, Logistikzentrum	50000	Erneuerbar	MALL	MALL	MULL
107	3D-Drucker HighTech	10	150	1	MALL	MALL		8	Dresden, Vertriebszentrum	60000	Erneuerbar	MALL	MALL	MULL
108	Nessus Produkt	0.6	250	1	MALL	MALL		1	Berlin, Lagerstraße 12	56000	Solar	MALL	MALL	MULL
108	Nessus Produkt	0.6	250	1	MALL	MALL		2	Hamburg, Hafenlager 23	75000	Wind	MALL	MALL	MULL
108	Nessus Produkt	0.6	250	1	MALL	MALL		3	München, Südalles 45	35000	Solar	MALL	MALL	MULL
108	Nessus Produkt	0.6	250	1	MALL	MALL		4	Frankfurt, Westhafen 8	100000	Solar	MALL	MALL	MULL
108	Nessus Produkt	0.6	250	1	MALL	MALL		5	Leipzig, Zentrallager	45000	Erneuerbar	MALL	MALL	MULL
108	Nessus Produkt	0.6	250	1	MALL	MALL		6	Stuttgart, Hauptlager 1	55000	Erneuerbar	MALL	MALL	MULL
108	Nessus Produkt	0.6	250	1	MALL	MALL		7	Düsseldorf, Logistikzentrum	50000	Erneuerbar	MALL	MALL	MULL
108	Nessus Produkt	0.6	250	1	MALL	MALL		8	Dresden, Vertriebszentrum	60000	Erneuerbar	MALL	MALL	MULL
109	Smartwatch Pro	0.36	450	1	MALL	MALL		1	Berlin, Lagerstraße 12	56000	Solar	MALL	MALL	MULL
109	Smartwatch Pro	0.36	450	1	MALL	MALL		2	Hamburg, Hafenlager 23	75000	Wind	MALL	MALL	MULL
109	Smartwatch Pro	0.36	450	1	MALL	MALL		3	München, Südalles 45	35000	Solar	MALL	MALL	MULL
109	Smartwatch Pro	0.36	450	1	MALL	MALL		4	Frankfurt, Westhafen 8	100000	Solar	MALL	MALL	MULL
109	Smartwatch Pro	0.36	450	1	MALL	MALL		5	Leipzig, Zentrallager	45000	Erneuerbar	MALL	MALL	MULL
109	Smartwatch Pro	0.36	450	1	MALL	MALL		6	Stuttgart, Hauptlager 1	55000	Erneuerbar	MALL	MALL	MULL
109	Smartwatch Pro	0.36	450	1	MALL	MALL		7	Düsseldorf, Logistikzentrum	50000	Erneuerbar	MALL	MALL	MULL
109	Smartwatch Pro	0.36	450	1	MALL	MALL		8	Dresden, Vertriebszentrum	60000	Erneuerbar	MALL	MALL	MULL

CROSS JOIN (Kartesisches Produkt)

Ein CROSS JOIN erstellt alle möglichen Kombinationen von zwei Tabellen

Alle Kombinationen von Fahrzeugen und Lieferungen

SELECT F.Fahrzeug_ID, F.Typ, L.Lieferungsnummer, L.Zielort

FROM Fahrzeug AS F

CROSS JOIN Lieferung AS L;

Die obige Operation erstellt jede mögliche Fahrzeug-Lieferungs-Kombination, unabhängig davon, ob sie tatsächlich existiert.

Fahrzeug_ID	Typ	Lieferungsnummer	Zielort
1	Elektro-LKM	1	Berlin, Alexanderplatz
3	Elektro-Van	1	Berlin, Alexanderplatz
5	Autonomer LKM	1	Berlin, Alexanderplatz
6	Wasserstoff-LKM	1	Berlin, Alexanderplatz
9	Elektro-LKM	1	Berlin, Alexanderplatz
10	MULL	1	Berlin, Alexanderplatz
11	Hybrid-LKM	1	Berlin, Alexanderplatz
12	Drohne	1	Berlin, Alexanderplatz
13	Elektro-Van	1	Berlin, Alexanderplatz
14	MULL	1	Berlin, Alexanderplatz
15	Autonomer LKM	1	Berlin, Alexanderplatz
16	Wasserstoff-LKM	1	Berlin, Alexanderplatz
17	MULL	1	Berlin, Alexanderplatz
18	Hybrid-Drohne	2	Berlin, Alexanderplatz
1	Elektro-LKM	2	Hamburg, Hafenstraße
3	Elektro-Van	2	Hamburg, Hafenstraße
5	Autonomer LKM	2	Hamburg, Hafenstraße
6	Wasserstoff-LKM	2	Hamburg, Hafenstraße
9	Elektro-LKM	2	Hamburg, Hafenstraße
10	MULL	2	Hamburg, Hafenstraße
11	Hybrid-LKM	2	Hamburg, Hafenstraße
12	Drohne	2	Hamburg, Hafenstraße
13	Elektro-Van	2	Hamburg, Hafenstraße
14	MULL	2	Hamburg, Hafenstraße
15	Autonomer LKM	2	Hamburg, Hafenstraße
16	Wasserstoff-LKM	2	Hamburg, Hafenstraße
17	MULL	2	Hamburg, Hafenstraße
18	Hybrid-Drohne	2	Hamburg, Hafenstraße
1	Elektro-LKM	3	München, Marienplatz
3	Elektro-Van	3	München, Marienplatz
5	Autonomer LKM	3	München, Marienplatz
6	Wasserstoff-LKM	3	München, Marienplatz
9	Elektro-LKM	3	München, Marienplatz
10	MULL	3	München, Marienplatz
11	Hybrid-LKM	3	München, Marienplatz
12	Drohne	3	München, Marienplatz
13	Elektro-Van	3	München, Marienplatz
14	MULL	3	München, Marienplatz
15	Autonomer LKM	3	München, Marienplatz
16	Wasserstoff-LKM	3	München, Marienplatz
17	MULL	3	München, Marienplatz
18	Hybrid-Drohne	3	München, Marienplatz
1	Elektro-LKM	4	Frankfurt, Zeilstraße
3	Elektro-Van	4	Frankfurt, Zeilstraße
5	Autonomer LKM	4	Frankfurt, Zeilstraße
6	Wasserstoff-LKM	4	Frankfurt, Zeilstraße
9	Elektro-LKM	4	Frankfurt, Zeilstraße
10	MULL	4	Frankfurt, Zeilstraße
11	Hybrid-LKM	4	Frankfurt, Zeilstraße
12	Drohne	4	Frankfurt, Zeilstraße
13	Elektro-Van	4	Frankfurt, Zeilstraße
14	MULL	4	Frankfurt, Zeilstraße
15	Autonomer LKM	4	Frankfurt, Zeilstraße
16	Wasserstoff-LKM	4	Frankfurt, Zeilstraße
17	MULL	4	Frankfurt, Zeilstraße
18	Hybrid-Drohne	4	Frankfurt, Zeilstraße
1	Elektro-LKM	5	Unbekannt, Zeilstraße
3	Elektro-Van	5	Unbekannt, Zeilstraße
5	Autonomer LKM	5	Unbekannt, Zeilstraße
6	Wasserstoff-LKM	5	Unbekannt, Zeilstraße
9	Elektro-LKM	5	Unbekannt, Zeilstraße
10	MULL	5	Unbekannt, Zeilstraße
11	Hybrid-LKM	5	Unbekannt, Zeilstraße
12	Drohne	5	Unbekannt, Zeilstraße
13	Elektro-Van	5	Unbekannt, Zeilstraße
14	MULL	5	Unbekannt, Zeilstraße
15	Autonomer LKM	5	Unbekannt, Zeilstraße
16	Wasserstoff-LKM	5	Unbekannt, Zeilstraße
17	MULL	5	Unbekannt, Zeilstraße
18	Hybrid-Drohne	5	Unbekannt, Zeilstraße
1	Elektro-LKM	6	Unbekannt, Marienplatz
3	Elektro-Van	6	Unbekannt, Marienplatz
5	Autonomer LKM	6	Unbekannt, Marienplatz
6	Wasserstoff-LKM	6	Unbekannt, Marienplatz
9	Elektro-LKM	6	Unbekannt, Marienplatz
10	MULL	6	Unbekannt, Marienplatz
11	Hybrid-LKM	6	Unbekannt, Marienplatz
12	Drohne	6	Unbekannt, Marienplatz
13	Elektro-Van	6	Unbekannt, Marienplatz
14	MULL	6	Unbekannt, Marienplatz
15	Autonomer LKM	6	Unbekannt, Marienplatz
16	Wasserstoff-LKM	6	Unbekannt, Marienplatz
17	MULL	6	Unbekannt, Marienplatz
18	Hybrid-Drohne	6	Unbekannt, Marienplatz
1	Elektro-LKM	7	Unbekannt, Marienplatz
3	Elektro-Van	7	Unbekannt, Marienplatz
5	Autonomer LKM	7	Unbekannt, Marienplatz
6	Wasserstoff-LKM	7	Unbekannt, Marienplatz
9	Elektro-LKM	7	Unbekannt, Marienplatz
10	MULL	7	Unbekannt, Marienplatz
11	Hybrid-LKM	7	Unbekannt, Marienplatz
12	Drohne	7	Unbekannt, Marienplatz
13	Elektro-Van	7	Unbekannt, Marienplatz
14	MULL	7	Unbekannt, Marienplatz
15	Autonomer LKM	7	Unbekannt, Marienplatz
16	Wasserstoff-LKM	7	Unbekannt, Marienplatz
17	MULL	7	Unbekannt, Marienplatz
18	Hybrid-Drohne	7	Unbekannt, Marienplatz
1	Elektro-LKM	8	Unbekannt, Marienplatz
3	Elektro-Van	8	Unbekannt, Marienplatz
5	Autonomer LKM	8	Unbekannt, Marienplatz
6	Wasserstoff-LKM	8	Unbekannt, Marienplatz
9	Elektro-LKM	8	Unbekannt, Marienplatz
10	MULL	8	Unbekannt, Marienplatz
11	Hybrid-LKM	8	Unbekannt, Marienplatz
12	Drohne	8	Unbekannt, Marienplatz
13	Elektro-Van	8	Unbekannt, Marienplatz
14	MULL	8	Unbekannt, Marienplatz
15	Autonomer LKM	8	Unbekannt, Marienplatz
16	Wasserstoff-LKM	8	Unbekannt, Marienplatz
17	MULL	8	Unbekannt, Marienplatz
18	Hybrid-Drohne	8	Unbekannt, Marienplatz
1	Elektro-LKM	9	Köln, Domplatz
3	Elektro-Van	9	Köln, Domplatz
5	Autonomer LKM	9	Köln, Domplatz
6	Wasserstoff-LKM	9	Köln, Domplatz
9	Elektro-LKM	9	Köln, Domplatz
10	MULL	9	Köln, Domplatz
11	Hybrid-LKM	9	Köln, Domplatz
12	Drohne	9	Köln, Domplatz
13	Elektro-Van	9	Köln, Domplatz
14	MULL	9	Köln, Domplatz
15	Autonomer LKM	9	Köln, Domplatz
16	Wasserstoff-LKM	9	Köln, Domplatz
17	MULL	9	Köln, Domplatz
18	Hybrid-Drohne	9	Köln, Domplatz
1	Elektro-LKM	18	MULL
3	Elektro-Van	18	MULL
5	Autonomer LKM	18	MULL
6	Wasserstoff-LKM	18	MULL
9	Elektro-LKM	18	MULL
10	MULL	18	MULL
11	Hybrid-LKM	18	MULL
12	Drohne	18	MULL
13	Elektro-Van	18	MULL
14	MULL	18	MULL
15	Autonomer LKM	18	MULL
16	Wasserstoff-LKM	18	MULL
17	MULL	18	MULL
18	Hybrid-Drohne	18	MULL
1	Elektro-LKM	11	Hamburg, Reeperbahn
3	Elektro-Van	11	Hamburg, Reeperbahn
5	Autonomer LKM	11	Hamburg, Reeperbahn
6	Wasserstoff-LKM	11	Hamburg, Reeperbahn
9	Elektro-LKM	11	Hamburg, Reeperbahn
10	MULL	11	Hamburg, Reeperbahn
11	Hybrid-LKM	11	Hamburg, Reeperbahn
12	Drohne	11	Hamburg, Reeperbahn
13	Elektro-Van	11	Hamburg, Reeperbahn
14	MULL	11	Hamburg, Reeperbahn
15	Autonomer LKM	11	Hamburg, Reeperbahn
16	Wasserstoff-LKM	11	Hamburg, Reeperbahn
17	MULL	11	Hamburg, Reeperbahn
18	Hybrid-Drohne	11	Hamburg, Reeperbahn

Das ganze Ergebnis ist zu lang

Alle Artikel mit allen Lagerhäusern kombinieren

SELECT A.Artikelnummer, A.Bezeichnung, L.Lager_ID, L.Standort

FROM Artikel AS A

CROSS JOIN Lagerhaus AS L;

Die obige Operation kombiniert jeden Artikel mit jedem Lagerhaus, um potenzielle Lageroptionen zu analysieren.

```
MariaDB [hosakbari_db]> SELECT A.Artikelnummer, A.Bezeichnung, L.Lager_ID, L.Standort
→ FROM Artikel AS A
→ CROSS JOIN Lagerhaus AS L;
```

Artikelnummer	Bezeichnung	Lager_ID	Standort
101	Smartphone-Modell X	1	Berlin, Lagerstraße 12
101	Smartphone-Modell X	2	Hamburg, Hafenlager 23
101	Smartphone-Modell X	3	München, Südallee 45
101	Smartphone-Modell X	4	Frankfurt, Westhafen 8
101	Smartphone-Modell X	5	Leipzig, Zentrallager
101	Smartphone-Modell X	6	Stuttgart, Hauptlager 1
101	Smartphone-Modell X	7	Düsseldorf, Logistikzentrum
101	Smartphone-Modell X	8	Dresden, Verteilzentrum
102	Laptop-Modell Y	1	Berlin, Lagerstraße 12
102	Laptop-Modell Y	2	Hamburg, Hafenlager 23
102	Laptop-Modell Y	3	München, Südallee 45
102	Laptop-Modell Y	4	Frankfurt, Westhafen 8
102	Laptop-Modell Y	5	Leipzig, Zentrallager
102	Laptop-Modell Y	6	Stuttgart, Hauptlager 1
102	Laptop-Modell Y	7	Düsseldorf, Logistikzentrum
102	Laptop-Modell Y	8	Dresden, Verteilzentrum
103	Tablet-Modell Z	1	Berlin, Lagerstraße 12
103	Tablet-Modell Z	2	Hamburg, Hafenlager 23
103	Tablet-Modell Z	3	München, Südallee 45
103	Tablet-Modell Z	4	Frankfurt, Westhafen 8
103	Tablet-Modell Z	5	Leipzig, Zentrallager
103	Tablet-Modell Z	6	Stuttgart, Hauptlager 1
103	Tablet-Modell Z	7	Düsseldorf, Logistikzentrum
103	Tablet-Modell Z	8	Dresden, Verteilzentrum
104	Kühlschrank Eco-Kühl	1	Berlin, Lagerstraße 12
104	Kühlschrank Eco-Kühl	2	Hamburg, Hafenlager 23
104	Kühlschrank Eco-Kühl	3	München, Südallee 45
104	Kühlschrank Eco-Kühl	4	Frankfurt, Westhafen 8
104	Kühlschrank Eco-Kühl	5	Leipzig, Zentrallager
104	Kühlschrank Eco-Kühl	6	Stuttgart, Hauptlager 1
104	Kühlschrank Eco-Kühl	7	Düsseldorf, Logistikzentrum
104	Kühlschrank Eco-Kühl	8	Dresden, Verteilzentrum
105	Waschmaschine SuperClean	1	Berlin, Lagerstraße 12
105	Waschmaschine SuperClean	2	Hamburg, Hafenlager 23
105	Waschmaschine SuperClean	3	München, Südallee 45
105	Waschmaschine SuperClean	4	Frankfurt, Westhafen 8
105	Waschmaschine SuperClean	5	Leipzig, Zentrallager
105	Waschmaschine SuperClean	6	Stuttgart, Hauptlager 1
105	Waschmaschine SuperClean	7	Düsseldorf, Logistikzentrum
105	Waschmaschine SuperClean	8	Dresden, Verteilzentrum
106	Drucker Modell T100	1	Berlin, Lagerstraße 12
106	Drucker Modell T100	2	Hamburg, Hafenlager 23
106	Drucker Modell T100	3	München, Südallee 45
106	Drucker Modell T100	4	Frankfurt, Westhafen 8
106	Drucker Modell T100	5	Leipzig, Zentrallager
106	Drucker Modell T100	6	Stuttgart, Hauptlager 1
106	Drucker Modell T100	7	Düsseldorf, Logistikzentrum
106	Drucker Modell T100	8	Dresden, Verteilzentrum
107	3D-Drucker HighTech	1	Berlin, Lagerstraße 12
107	3D-Drucker HighTech	2	Hamburg, Hafenlager 23
107	3D-Drucker HighTech	3	München, Südallee 45
107	3D-Drucker HighTech	4	Frankfurt, Westhafen 8
107	3D-Drucker HighTech	5	Leipzig, Zentrallager
107	3D-Drucker HighTech	6	Stuttgart, Hauptlager 1
107	3D-Drucker HighTech	7	Düsseldorf, Logistikzentrum
107	3D-Drucker HighTech	8	Dresden, Verteilzentrum
108	Neues Produkt	1	Berlin, Lagerstraße 12
108	Neues Produkt	2	Hamburg, Hafenlager 23
108	Neues Produkt	3	München, Südallee 45
108	Neues Produkt	4	Frankfurt, Westhafen 8
108	Neues Produkt	5	Leipzig, Zentrallager
108	Neues Produkt	6	Stuttgart, Hauptlager 1
108	Neues Produkt	7	Düsseldorf, Logistikzentrum
108	Neues Produkt	8	Dresden, Verteilzentrum
109	Smartwatch Pro	1	Berlin, Lagerstraße 12
109	Smartwatch Pro	2	Hamburg, Hafenlager 23
109	Smartwatch Pro	3	München, Südallee 45
109	Smartwatch Pro	4	Frankfurt, Westhafen 8
109	Smartwatch Pro	5	Leipzig, Zentrallager
109	Smartwatch Pro	6	Stuttgart, Hauptlager 1
109	Smartwatch Pro	7	Düsseldorf, Logistikzentrum
109	Smartwatch Pro	8	Dresden, Verteilzentrum

72 rows in set (0,001 sec)

AS (Alias für Tabellen oder Spalten)

AS gibt Tabellen oder Spalten kürzere Namen für bessere Lesbarkeit

Alias für Tabellen und Spaltennamen

```
SELECT L.Lieferungsnummer AS ID, L.Zielort AS Ziel, F.Typ AS Fahrzeugtyp
FROM Lieferung AS L
INNER JOIN Fahrzeug AS F ON L.Lieferungsnummer = F.Lieferung_Lieferungsnummer;
```

Die obige Operation verkürzt die Spaltennamen (Lieferungsnummer -> ID, Zielort -> Ziel). Durch L und F werden Abkürzungen für die Tabellen erzeugt, um sie kürzer zu schreiben.

```
MariaDB [hosakbari_db]> SELECT L.Lieferungsnummer AS ID, L.Zielort AS Ziel, F.Typ AS Fahrzeugtyp
    ER JOIN Fahrzeug      → FROM Lieferung AS L
        → INNER JOIN Fahrzeug AS F ON L.Lieferungsnummer = F.Lieferung_Lieferungsnummer;
+----+-----+-----+
| ID | Ziel          | Fahrzeugtyp |
+----+-----+-----+
| 1  | Berlin, Alexanderplatz | Elektro-LKW |
| 3  | München, Marienplatz   | Elektro-Van  |
| 3  | München, Marienplatz   | Autonomer LKW |
| 3  | München, Marienplatz   | Wasserstoff-LKW |
| 1  | Berlin, Alexanderplatz | NULL         |
| 3  | München, Marienplatz   | Hybrid-LKW  |
| 4  | Frankfurt, Zeilstraße   | Drohne       |
| 2  | Hamburg, Hafenstraße   | Autonomer LKW |
| 5  | Unbekannt               | Wasserstoff-LKW |
+----+-----+-----+
9 rows in set (0,001 sec)
```

Alias für Berechnungen

```
SELECT Artikelnummer, Bezeichnung, Lagerbestand * 2 AS Doppelte_Menge
FROM Artikel;
```

Oben wird die doppelte Menge berechnet und als Doppelte_Menge angezeigt.

```
MariaDB [hosakbari_db]> SELECT Artikelnummer, Bezeichnung, Lagerbestand * 2 AS Doppelte_Menge
    → FROM Artikel;
+-----+-----+-----+
| Artikelnummer | Bezeichnung | Doppelte_Menge |
+-----+-----+-----+
| 101 | Smartphone-Modell X | 2900 |
| 102 | Laptop-Modell Y | 900  |
| 103 | Tablet-Modell Z | 1500 |
| 104 | Kühlschrank Eco-Kühl | 140  |
| 105 | Waschmaschine SuperClean | 100 |
| 106 | Drucker Modell T100 | 200  |
| 107 | 3D-Drucker HighTech | 300  |
| 108 | Neues Produkt | 500  |
| 109 | Smartwatch Pro | 900  |
+-----+-----+-----+
9 rows in set (0,001 sec)
```

LEFT JOIN (Linker äußerer Verbund)

Ein LEFT JOIN zeigt alle Zeilen der linken Tabelle, auch wenn in der rechten Tabelle keine Übereinstimmungen existiert.

```
SELECT L.Lieferungsnummer, L.Zielort, F.Typ AS Fahrzeugtyp
FROM Lieferung AS L
LEFT JOIN Fahrzeug AS F ON L.Lieferungsnummer = F.Lieferung_Lieferungsnummer;
```

Oben werden alle Lieferungen angezeigt, auch wenn kein Fahrzeug zugewiesen ist (dann steht NULL in Fahrzeugtyp).

```
MariaDB [hosakbari_db]> SELECT L.Lieferungsnummer, L.Zielort, F.Typ AS Fahrzeugtyp
    → FROM Lieferung AS L
    → LEFT JOIN Fahrzeug AS F ON L.Lieferungsnummer = F.Lieferung_Lieferungsnummer;
+-----+-----+-----+
| Lieferungsnummer | Zielort | Fahrzeugtyp |
+-----+-----+-----+
| 1 | Berlin, Alexanderplatz | Elektro-LKW |
| 1 | Berlin, Alexanderplatz | NULL |
| 2 | Hamburg, Hafenstraße | Autonomer LKW |
| 3 | München, Marienplatz | Elektro-Van |
| 3 | München, Marienplatz | Autonomer LKW |
| 3 | München, Marienplatz | Wasserstoff-LKW |
| 3 | München, Marienplatz | Hybrid-LKW |
| 4 | Frankfurt, Zeilstraße | Drohne |
| 5 | Unbekannt | Wasserstoff-LKW |
| 6 | Unbekannt | NULL |
| 7 | Unbekannt | NULL |
| 8 | Unbekannt | NULL |
| 9 | Köln, Domplatz | NULL |
| 10 | NULL | NULL |
| 11 | Hamburg, Reeperbahn | NULL |
| 12 | München, Stachus | NULL |
| 13 | NULL | NULL |
| 14 | Düsseldorf, Altstadt | NULL |
| 15 | Stuttgart, Schlossplatz | NULL |
| 16 | Frankfurt, Mainufer | NULL |
| 17 | NULL | NULL |
| 18 | Berlin, Kudamm | NULL |
| 19 | NULL | NULL |
| 20 | Nürnberg, Zentrum | NULL |
| 21 | Bremen, Innenstadt | NULL |
| 22 | NULL | NULL |
| 23 | Hannover, Hauptbahnhof | NULL |
| 24 | NULL | NULL |
+-----+-----+-----+
```

28 rows in set (0,030 sec)

Alle Artikel mit ihrem Lagerhaus (falls vorhanden)

```
SELECT A.Artikelnummer, A.Bezeichnung, L.Standort
FROM Artikel AS A
LEFT JOIN Lagerhaus AS L ON A.Lagerhaus_Lager_ID = L.Lager_ID;
```

Oben werden alle Artikel gezeigt, auch wenn kein Lagerhaus zugewiesen werden kann.

```
MariaDB [hosakbari_db]> SELECT A.Artikelnummer, A.Bezeichnung, L.Standort
    → FROM Artikel AS A
    → LEFT JOIN Lagerhaus AS L ON A.Lagerhaus_Lager_ID = L.Lager_ID;
+-----+-----+-----+
| Artikelnummer | Bezeichnung | Standort |
+-----+-----+-----+
| 101 | Smartphone-Modell X | Berlin, Lagerstraße 12 |
| 102 | Laptop-Modell Y | Hamburg, Hafenlager 23 |
| 103 | Tablet-Modell Z | München, Südallee 45 |
| 104 | Kühlschrank Eco-Kühl | Frankfurt, Westhafen 8 |
| 105 | Waschmaschine SuperClean | Frankfurt, Westhafen 8 |
| 106 | Drucker Modell T100 | Berlin, Lagerstraße 12 |
| 107 | 3D-Drucker HighTech | Berlin, Lagerstraße 12 |
| 108 | Neues Produkt | Berlin, Lagerstraße 12 |
| 109 | Smartwatch Pro | Berlin, Lagerstraße 12 |
+-----+-----+-----+
```

9 rows in set (0,002 sec)

RIGHT OUTER JOIN / RIGHT JOIN

Ein RIGHT JOIN zeigt alle Zeilen der rechten Tabelle, selbst wenn keine Übereinstimmung in der linken Tabelle existiert.

Alle Lagerhäuser mit Artikeln (falls vorhanden)

```
SELECT A.Artikelnummer, A.Bezeichnung, L.Lager_ID, L.Standort
FROM Artikel AS A
RIGHT JOIN Lagerhaus AS L ON A.Lagerhaus_Lager_ID = L.Lager_ID;
```

Oben werden alle Lagerhäuser gezeigt, selbst wenn sie keine Artikel enthalten (NULL in Artikelnummer).

```
MariaDB [hosakbari_db]> SELECT A.Artikelnummer, A.Bezeichnung, L.Lager_ID, L.Standort
→ FROM Artikel AS A
→ RIGHT JOIN Lagerhaus AS L ON A.Lagerhaus_Lager_ID = L.Lager_ID;
```

Artikelnummer	Bezeichnung	Lager_ID	Standort
101	Smartphone-Modell X	1	Berlin, Lagerstraße 12
106	Drucker Modell T100	1	Berlin, Lagerstraße 12
107	3D-Drucker HighTech	1	Berlin, Lagerstraße 12
108	Neues Produkt	1	Berlin, Lagerstraße 12
109	Smartwatch Pro	1	Berlin, Lagerstraße 12
102	Laptop-Modell Y	2	Hamburg, Hafenlager 23
103	Tablet-Modell Z	3	München, Südallee 45
104	Kühlschrank Eco-Kühl	4	Frankfurt, Westhafen 8
105	Waschmaschine SuperClean	4	Frankfurt, Westhafen 8
NULL	NULL	5	Leipzig, Zentrallager
NULL	NULL	6	Stuttgart, Hauptlager 1
NULL	NULL	7	Düsseldorf, Logistikzentrum
NULL	NULL	8	Dresden, Verteilzentrum

13 rows in set (0,026 sec)

Alle Lieferungen mit zugewiesenen Artikeln (falls vorhanden)

```
SELECT L.Lieferungsnummer, L.Zielort, A.Bezeichnung
FROM Lieferung AS L
RIGHT JOIN Relation_Lieferung_Artikel AS R ON L.Lieferungsnummer = R.Lieferungsnummer
RIGHT JOIN Artikel AS A ON R.Artikelnummer = A.Artikelnummer;
```

Oben werden alle Artikel gezeigt, selbst wenn sie keiner Lieferung zugeordnet sind.

```
MariaDB [hosakbari_db]> SELECT L.Lieferungsnummer, L.Zielort, A.Bezeichnung
→ FROM Lieferung AS L
→ RIGHT JOIN Relation_Lieferung_Artikel AS R ON L.Lieferungsnummer = R.Lieferungsnummer
→ RIGHT JOIN Artikel AS A ON R.Artikelnummer = A.Artikelnummer;
```

Lieferungsnummer	Zielort	Bezeichnung
1	Berlin, Alexanderplatz	Smartphone-Modell X
4	Frankfurt, Zeilstraße	Smartphone-Modell X
NULL	NULL	Laptop-Modell Y
2	Hamburg, Hafenstraße	Tablet-Modell Z
4	Frankfurt, Zeilstraße	Tablet-Modell Z
NULL	NULL	Kühlschrank Eco-Kühl
NULL	NULL	Waschmaschine SuperClean
NULL	NULL	Drucker Modell T100
NULL	NULL	3D-Drucker HighTech
NULL	NULL	Neues Produkt
NULL	NULL	Smartwatch Pro

11 rows in set (0,013 sec)

Normalisieren Sie Relationen bis zur 3. Normalform.

Bei dieser Aufgabenstellung geht es darum die Tabellen bis zur dritten Normalform zu normalisieren. Bei der ersten Normalform (1NF) hat man das Ziel die Tabelle so umzugestalten, dass jede Zelle einer Tabelle nur einen einzigen, atomaren Wert enthalten darf. Hierzu dürfen keine mehrfachen Werte oder Listen in einer Spalte vorkommen.

Beispiel (nicht 1NF-konform)

Kunden-ID	Name	Telefonnummern
1	Alice	12345, 67890
2	Bob	54321

Lösung (1NF-konform): Jede Telefonnummer bekommt eine eigene Zeile.

Kunden-ID	Name	Telefonnummer
1	Alice	12345
1	Alice	67890
2	Bob	54321

Bei der zweiten Normalform muss man die Tabelle so umgestalten (vorausgesetzt die erste Normalform ist erfüllt), dass jedes Nicht-Schlüssel-Attribut vollständig vom Primärschlüssel abhängt, nicht nur von einem Teil davon.

Beispiel (nicht 2NF-konform): Tabelle „Bestellungen“

Bestell-ID	Produkt-ID	Produktnname	Kunde-ID	Kundenname
101	P1	Laptop	1	Alice
102	P2	Maus	2	Bob

Das Problem das wir haben ist es, dass Produktnname nur von Produkt-ID abhängt, nicht von Bestell-ID. Zudem hängt Kundenname nur von Kunde-ID ab.

Wir lösen das dadurch, dass wir das in separaten Tabellen aufteilen:

Tabelle „Bestellungen“

Bestell-ID	Produkt-ID	Kunde-ID
101	P1	1
102	P2	2

Tabelle „Produkte“

Produkt-ID	Produktnname
P1	Laptop
P2	Maus

Tabelle „Kunden“:

Kunde-ID	Kundenname
1	Alice
2	Bob

Bei der dritten Normalform (3NF) darf es in einer Tabelle keine transitiven Abhängigkeiten geben. Zudem darf ein Nicht-Schlüssel-Attribut nicht von einem anderen Nicht-Schlüssel-Attribut abhängen. Voraussetzung für die dritte Normalform, ist die zweite Normalform.

Beispiel (nicht 3NF-konform): Tabelle „Kunden“

Kunde-ID	Kundenname	PLZ	Stadt
1	Alice	12345	Berlin
2	Bob	67890	Hamburg

Das Problem das wir hierbei haben ist der, dass die Stadt von PLZ abhängt, nicht direkt von Kunde-ID (transitive Abhängigkeit).

Lösung (3NF-konform)

Tabelle „Kunden“:

Kunde-ID	Kundenname	PLZ
1	Alice	12345
2	Bob	67890

Tabelle „Postleitzahlen“:

PLZ	Stadt
12345	Berlin
67890	Hamburg

Als Fazit lässt sich sagen, dass bei der ersten Normalform keine mehrfachen Werte in einer Zelle vorhanden sein dürfen (atomare Werte), dass bei der zweiten Normalform es keine partiellen Abhängigkeiten von einem zusammengesetzten Primärschlüssel geben darf, sowie dass bei der dritten Normalform keine transitiven Abhängigkeiten (jedes Nicht-Schlüssel-Attribut hängt nur vom Primärschlüssel ab) geben darf. Hierdurch wird sichergestellt, dass die Datenbank redundanzfrei, konsistent und effizient bleibt.

Im Folgenden sollen alle vorhandenen Tabellen Artikel, Fahrzeug, Lagerhaus, Lieferung sowie Relation_Lieferung_Artikel bis zur dritten Normalform normalisiert werden. Hierzu fangen wir mit der ersten Tabelle Artikel an.

MariaDB [hosakbari_db]> SELECT * FROM Artikel;

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID	Produktionsland	Garantiezeit	Recyclingklasse
101	Smartphone-Modell X	0.24	1450	1	NULL	NULL	NULL
102	Laptop-Modell Y	1.8	450	2	NULL	NULL	NULL
103	Tablet-Modell Z	0.96	750	3	NULL	NULL	NULL
104	Kühlschrank Eco-Kühl	50	70	4	NULL	NULL	NULL
105	Waschmaschine SuperClean	75	50	4	NULL	NULL	NULL
106	Drucker Modell T100	5	100	1	NULL	NULL	NULL
107	3D-Drucker HighTech	10	150	1	NULL	NULL	NULL
108	Neues Produkt	0.6	250	1	NULL	NULL	NULL
109	Smartwatch Pro	0.36	450	1	NULL	NULL	NULL

9 rows in set (0,002 sec)

Hierzu soll die Tabelle Artikel in die dritte Normalform überführt werden, ohne dass bereits vorhandene Daten verloren gehen. Dazu werden neue Tabellen erstellt, Abhängigkeiten analysiert und Daten migriert, während die Originaltabelle (oben zusehen) schrittweise angepasst wird.

Zuerst analysieren wir die aktuelle Struktur der Tabelle. Hierzu ist zu sagen, dass die Tabelle bereits die 1NF erfüllt, da alle Attribute atomar sind. Die 2NF jedoch ist nicht erfüllt, da funktionale Abhängigkeiten von Nicht-Schlüsselattributen existieren. Wegen des Vorhandenseins von transitiven Abhängigkeiten ist somit die 3NF ebenfalls nicht erfüllt.

Nun geht es darum die Tabelle auf die zweite Normalform umzustrukturieren. Die Problematik in der 2NF liegt darin, dass Produktionsland, Garantiezeit und Recyclingklasse nicht direkt von der Artikelnummer abhängen, sondern eher von einer Art „Produktyp“ (z.B. Smartphone, Laptop, Kühlschrank). Eine zweite Problematik ergibt sich aufgrund von Redundanz, da mehrere Artikel desselben Typs die gleichen Werte für Produktionsland, Garantiezeit, sowie Recyclingklasse haben.

Um die Problematik zu lösen, trennen wir die Produkttyp-spezifischen Informationen in einer separaten Tabelle Produkttyp, die dann über eine neue Fremdschlüssel-Spalte Produkttyp_ID mit der Artikel-Tabelle verknüpft sind.

SQL-Schritt zur Umsetzung hiervon umfassen:

1. Neue Tabelle Produkttyp erstellen

```
CREATE TABLE Produkttyp (
    Produkttyp_ID INT AUTO_INCREMENT PRIMARY KEY,
    Bezeichnung VARCHAR(255) NOT NULL,
    Produktionsland VARCHAR(255),
    Garantiezeit INT,
    Recyclingklasse VARCHAR(50)
);
```

2. Produkttyp_ID in die Tabelle Artikel aufnehmen

```
ALTER TABLE Artikel ADD COLUMN Produkttyp_ID INT;
ALTER TABLE Artikel ADD FOREIGN KEY (Produkttyp_ID) REFERENCES Produkttyp(Produkttyp_ID);
```

3. Daten aus Artikel in Produkttyp übertragen

```
INSERT INTO Produkttyp (Bezeichnung, Produktionsland, Garantiezeit, Recyclingklasse)
SELECT DISTINCT Bezeichnung, Produktionsland, Garantiezeit, Recyclingklasse FROM Artikel;
```

4. Produkttyp_ID in Artikel füllen

```
UPDATE Artikel a
JOIN Produkttyp p ON a.Bezeichnung = p.Bezeichnung
SET a.Producekttyp_ID = p.Producekttyp_ID;
```

5. Redundante Spalten aus Artikel entfernen

```
ALTER TABLE Artikel DROP COLUMN Produktionsland;
ALTER TABLE Artikel DROP COLUMN Garantiezeit;
ALTER TABLE Artikel DROP COLUMN Recyclingklasse;
```

MariaDB [hosakbari_db]> SELECT * FROM Artikel;

Artikelnummer	Bezeichnung	Gewicht	Lagerbestand	Lagerhaus_Lager_ID	Produkttyp_ID
101	Smartphone-Modell X	0.24	1450	1	1
102	Laptop-Modell Y	1.8	450	2	2
103	Tablet-Modell Z	0.96	750	3	3
104	Kühlschrank Eco-Kühl	50	70	4	4
105	Waschmaschine SuperClean	75	50	4	5
106	Drucker Modell T100	5	100	1	6
107	3D-Drucker HighTech	10	150	1	7
108	Neues Produkt	0.6	250	1	8
109	Smartwatch Pro	0.36	450	1	9

9 rows in set (0,002 sec)

MariaDB [hosakbari_db]> SELECT * FROM Produkttyp;

Produkttyp_ID	Bezeichnung	Produktionsland	Garantiezeit	Recyclingklasse
1	Smartphone-Modell X	NULL	NULL	NULL
2	Laptop-Modell Y	NULL	NULL	NULL
3	Tablet-Modell Z	NULL	NULL	NULL
4	Kühlschrank Eco-Kühl	NULL	NULL	NULL
5	Waschmaschine SuperClean	NULL	NULL	NULL
6	Drucker Modell T100	NULL	NULL	NULL
7	3D-Drucker HighTech	NULL	NULL	NULL
8	Neues Produkt	NULL	NULL	NULL
9	Smartwatch Pro	NULL	NULL	NULL

9 rows in set (0,001 sec)

Wie man aus dem Ergebnis sehen kann, ist der Artikel nun in der 2. Normalform (2NF), da sich alle Attribute direkt auf die Artikelnummer beziehen. Produktionsland, Garantiezeit, sowie Recyclingklasse sind nun in Produkttyp ausgelagert, sodass Redundanzen vermieden werden.

Nun geht es darum die Umstrukturierung auf die 3NF umzusetzen. Die Problematik in der 3NF ist der, dass ein Artikel zu einem Lagerhaus gehört, aber Lagerbestand sowohl vom Lagerhaus, als auch vom Artikel abhängt, und nicht nur vom Artikel.

Zur Lösung hierzu muss die Lagerbestand-Information in eine separate Tabelle ausgelagert werden, da sie eine m:n-Beziehung zwischen Artikel und Lagerhaus darstellt.

Erneut ergeben sich zur Umsetzung hiervon SQL-Schritte wie folgt:

1. Neue Tabelle Lagerbestand erstellen

```
CREATE TABLE Lagerbestand (
    Artikelnummer INT,
    Lager_ID INT,
    Bestand INT,
    PRIMARY KEY (Artikelnummer, Lager_ID),
    FOREIGN KEY (Artikelnummer) REFERENCES Artikel(Artikelnummer),
    FOREIGN KEY (Lager_ID) REFERENCES Lagerhaus(Lager_ID)
);
```

2. Daten aus Artikel in Lagerbestand übertragen

```
INSERT INTO Lagerbestand (Artikelnummer, Lager_ID, Bestand)
SELECT Artikelnummer, Lagerhaus_Lager_ID, Lagerbestand FROM Artikel;
```

3. Lagerhaus_Lager_ID und Lagerbestand aus Artikel entfernen

```
ALTER TABLE Artikel DROP COLUMN Lagerhaus_Lager_ID;
ALTER TABLE Artikel DROP COLUMN Lagerbestand;
```

```
MariaDB [hosakbari_db]> ALTER TABLE Artikel DROP COLUMN Lagerhaus_Lager_ID;
ERROR 1553 (HY000): Cannot drop index 'Lagerhaus_Lager_ID': needed in a foreign key constraint
MariaDB [hosakbari_db]> ALTER TABLE Artikel DROP COLUMN Lagerbestand;
Query OK, 0 rows affected (0,012 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Hierbei ist zusehen, dass durch den ERROR 1553 (HY000): Cannot drop index 'Lagerhaus_Lager_ID': needed in a foreign key constraint darauf hingewiesen wird, dass die Spalte Lagerhaus_Lager_ID in der Tabelle Artikel als Fremdschlüssel definiert wurde und daher nicht einfach gelöscht werden kann, solange diese Fremdschlüsselbeziehung existiert. Um dieses Problem zu lösen, müssen wir hierzu die bestehende Fremdschlüssel-Beschränkung identifizieren und die richtige Fremdschlüssel-Beschränkung entfernen. Für die Fremdschlüssel-Beschränkung-Identifizierung verwenden wir den Befehl SHOW CREATE TABLE Artikel;

```
MariaDB [hosakbari_db]> SHOW CREATE TABLE Artikel;
+-----+
| Table | Create Table
+-----+
| Artikel | CREATE TABLE `Artikel` (
  `Artikelnummer` int(11) NOT NULL,
  `Bezeichnung` varchar(300) DEFAULT NULL,
  `Gewicht` float DEFAULT NULL,
  `Lagerhaus_Lager_ID` int(11) DEFAULT NULL,
  `Produkttyp_ID` int(11) DEFAULT NULL,
  PRIMARY KEY (`Artikelnummer`),
  KEY `Lagerhaus_Lager_ID` (`Lagerhaus_Lager_ID`),
  KEY `Produkttyp_ID` (`Produkttyp_ID`),
  CONSTRAINT `Artikel_ibfk_1` FOREIGN KEY (`Lagerhaus_Lager_ID`) REFERENCES `Lagerhaus` (`Lager_ID`),
  CONSTRAINT `Artikel_ibfk_2` FOREIGN KEY (`Produkttyp_ID`) REFERENCES `Produkttyp` (`Produkttyp_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci |
+-----+
1 row in set (0,002 sec)
```

Wie man hierbei sehen kann, haben wir zwei Fremdschlüssel in der Artikel-Tabelle:

- Artikel_ibfk_1 bezieht sich auf Lagerhaus_Lager_ID → Lagerhaus(Lager_ID)
- Artikel_ibfk_2 bezieht sich auf Produkttyp_ID → Produkttyp(Produkttyp_ID)

Wir müssen hierzu den ersten Fremdschlüssel entfernen. Der zweite Fremdschlüssel wurde in der 2NF-Schritt erstellt und ist daher notwendig. Diese Spalte sollten wir also nicht löschen:

ALTER TABLE Artikel DROP FOREIGN KEY Artikel_ibfk_1;

Danach entfernen wir Lagerhaus_Lager_ID aus Artikel:

ALTER TABLE Artikel DROP COLUMN Lagerhaus_Lager_ID;

```
MariaDB [hosakbari_db]> SELECT * FROM Artikel;
+-----+
| Artikelnummer | Bezeichnung | Gewicht | Produkttyp_ID |
+-----+
| 101 | Smartphone-Modell X | 0.24 | 1 |
| 102 | Laptop-Modell Y | 1.8 | 2 |
| 103 | Tablet-Modell Z | 0.96 | 3 |
| 104 | Kühlschrank Eco-Kühl | 50 | 4 |
| 105 | Waschmaschine SuperClean | 75 | 5 |
| 106 | Drucker Modell T100 | 5 | 6 |
| 107 | 3D-Drucker HighTech | 10 | 7 |
| 108 | Neues Produkt | 0.6 | 8 |
| 109 | Smartwatch Pro | 0.36 | 9 |
+-----+
9 rows in set (0,004 sec)
```

Wie man anhand des Ergebnisses sehen kann, ist Artikel nun in der 3NF, weil alle Nicht-Schlüsselattribute direkt vom Primärschlüssel abhängen. Lagerbestand ist nun eine separate Tabelle, die eine m:n-Beziehung zwischen Artikel und Lagerhaus darstellt.

Das Ergebnis ist nun:

```
MariaDB [hosakbari_db]> SELECT * FROM Artikel;
```

Artikelnummer	Bezeichnung	Gewicht	Produkttyp_ID
101	Smartphone-Modell X	0.24	1
102	Laptop-Modell Y	1.8	2
103	Tablet-Modell Z	0.96	3
104	Kühlschrank Eco-Kühl	50	4
105	Waschmaschine SuperClean	75	5
106	Drucker Modell T100	5	6
107	3D-Drucker HighTech	10	7
108	Neues Produkt	0.6	8
109	Smartwatch Pro	0.36	9

9 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Produkttyp;
```

Produkttyp_ID	Bezeichnung	Produktionsland	Garantiezeit	Recyclingklasse
1	Smartphone-Modell X	NULL	NULL	NULL
2	Laptop-Modell Y	NULL	NULL	NULL
3	Tablet-Modell Z	NULL	NULL	NULL
4	Kühlschrank Eco-Kühl	NULL	NULL	NULL
5	Waschmaschine SuperClean	NULL	NULL	NULL
6	Drucker Modell T100	NULL	NULL	NULL
7	3D-Drucker HighTech	NULL	NULL	NULL
8	Neues Produkt	NULL	NULL	NULL
9	Smartwatch Pro	NULL	NULL	NULL

9 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerbestand;
```

Artikelnummer	Lager_ID	Bestand
101	1	1450
102	2	450
103	3	750
104	4	70
105	4	50
106	1	100
107	1	150
108	1	250
109	1	450

9 rows in set (0,001 sec)

Ergebnis dieser Normalisierung ist zum ersten, dass alle Attribute atomar sind, zum zweitens, dass alle Nicht-Schlüsselattribute direkt von der Artikelnummer abhängen, sowie drittens dass keine transitiven Abhängigkeiten mehr existieren (Produktionsland, Garantiezeit, Recyclingklasse ausgelagert). Weiterhin wurde darauf geachtet, dass keine Datenverlust entsteht, indem die bereits vorhandenen Daten in eine neue Tabelle übertragen würden.

Nun beginnen wir mit der Normalisierung der Tabelle Fahrzeug bis zur dritten Normalform. Die Normalisierung hierzu erfolgt in vier Schritten, um die dritte Normalform zu erreichen. Dabei wird erneut darauf geachtet, bestehende Daten zu erhalten, indem wir neue Tabellen erstellen und Daten schrittweise übertragen.

Im ersten Schritt analysieren wir die aktuelle Struktur der Tabelle Fahrzeug

Fahrzeug_ID	Typ	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer	Wartungsstatus	Ladezeit	CO2_Einsparung
1	Elektro-LKW	5000	85		1	NULL	NULL
3	Elektro-Van	1100	75		3	NULL	NULL
5	Autonomer LKW	7000	100		3	NULL	NULL
6	Wasserstoff-LKW	6500	100		3	NULL	NULL
9	Elektro-LKW	7000	90		NULL	NULL	NULL
10	NULL	1000	100		1	NULL	NULL
11	Hybrid-LKW	NULL	80		3	NULL	NULL
12	Drohne	50	NULL		4	NULL	NULL
13	Elektro-Van	2000	60		NULL	NULL	NULL
14	NULL	NULL	NULL		NULL	NULL	NULL
15	Autonomer LKW	7500	100		2	NULL	NULL
16	Wasserstoff-LKW	NULL	NULL		5	NULL	NULL
17	NULL	8000	75		NULL	NULL	NULL
18	Hybrid-Drohne	85	95		NULL	NULL	NULL

14 rows in set (0,001 sec)

Hierbei stellen wir folgende Problematiken fest: Mehrfach gespeicherte Fahrzeugtypen, was zu Datenredundanz führt, Transitive Abhängigkeit, da Fahrzeugtyp mehrere Attribute beeinflusst, sowie potenzielle NULL-Werte für einige Attribute.

Beim zweiten Schritt wird die Tabelle Fahrzeug in die 1NF umstrukturiert, da durch die mehrfache Speicherung desselben Fahrzeugtyps keine klar definierte Primärschlüsselstruktur existiert. Zur Lösung hiervon muss die Trennung der Fahrzeugtypen in eine eigene Tabelle Fahrzeugtyp erfolgen. Des Weiteren enthält jedes Attribut nur einen einzigen Wert, um die Atomizität zu gewährleisten.

Die SQL-Schritte hierzu umfassen:

1. Neue Tabelle Fahrzeugtyp erstellen

```
CREATE TABLE Fahrzeugtyp (
    Fahrzeugtyp_ID INT PRIMARY KEY AUTO_INCREMENT,
    Bezeichnung VARCHAR(255) NOT NULL
);
```

2. Fahrzeugtyp_ID in die Fahrzeug-Tabelle aufnehmen

```
ALTER TABLE Fahrzeug ADD COLUMN Fahrzeugtyp_ID INT;
```

3. Bestehende Fahrzeugtypen in die neue Tabelle Fahrzeugtyp einfügen

```
INSERT INTO Fahrzeugtyp (Bezeichnung)
```

```
SELECT DISTINCT Typ FROM Fahrzeug WHERE Typ IS NOT NULL;
```

4. Fahrzeugtyp_ID in die Fahrzeug-Tabelle übertragen

```
UPDATE Fahrzeug f
```

```
JOIN Fahrzeugtyp ft ON f.Typ = ft.Bezeichnung
```

```
SET f.Fahrzeugtyp_ID = ft.Fahrzeugtyp_ID;
```

5. Spalte Typ aus Fahrzeug entfernen

```
ALTER TABLE Fahrzeug DROP COLUMN Typ;
```

Fahrzeug_ID	Kapazität	Batteriestatus	Lieferung_Lieferungsnummer	Wartungsstatus	Ladezeit	CO2_Einsparung	Fahrzeugtyp_ID
1	5000	85		1	NULL	NULL	1
3	1100	75		3	NULL	NULL	2
5	7000	100		3	NULL	NULL	3
6	6500	100		3	NULL	NULL	4
9	7000	90		NULL	NULL	NULL	1
10	1000	100		1	NULL	NULL	NULL
11	NONE	80		3	NULL	NULL	5
12	50	NONE		4	NULL	NULL	6
13	2000	60		NULL	NULL	NULL	2
14	NONE	NONE		NULL	NULL	NULL	NULL
15	7500	100		2	NULL	NULL	3
16	NONE	NONE		5	NULL	NULL	4
17	8000	75		NULL	NULL	NULL	NULL
18	85	95		NULL	NULL	NULL	7

14 rows in set (0,002 sec)

MariaDB [hosakbari_db]> SELECT * FROM Fahrzeugtyp;

Fahrzeugtyp_ID	Bezeichnung
1	Elektro-LKW
2	Elektro-Van
3	Autonomer LKW
4	Wasserstoff-LKW
5	Hybrid-LKW
6	Drohne
7	Hybrid-Drohne

7 rows in set (0,001 sec)

So sieht das aktuelle Ergebnis aus, womit die 1NF sichergestellt wird.

Beim dritten Schritt sorgen wir für die Umstrukturierung in die 2NF. Die Problematik hierbei ist der, dass die Kapazität, Batteriestatus, Ladezeit und CO2_Einsparung vom Fahrzeugtyp abhängen, nicht vom Fahrzeug selbst. Diese Werte sollten deshalb in die Fahrzeugtyp-Tabelle verschoben werden. Zur Lösung hiervon werden die Spalten Kapazität, Batteriestatus, Ladezeit, CO2_Einsparung in die Tabelle Fahrzeugtyp verschoben.

Die Umsetzung hiervon erfolgt durch die folgenden SQL-Schritte:

1. Neue Spalten in Fahrzeugtyp hinzufügen

```
ALTER TABLE Fahrzeugtyp
ADD COLUMN Kapazität INT,
ADD COLUMN Batteriestatus INT,
ADD COLUMN Ladezeit INT,
ADD COLUMN CO2_Einsparung DECIMAL(10,2);
```

2. Werte aus Fahrzeug nach Fahrzeugtyp übertragen

```
UPDATE Fahrzeugtyp ft
JOIN Fahrzeug f ON ft.Fahrzeugtyp_ID = f.Fahrzeugtyp_ID
SET ft.Kapazität = f.Kapazität,
ft.Batteriestatus = f.Batteriestatus,
ft.Ladezeit = f.Ladezeit,
ft.CO2_Einsparung = f.CO2_Einsparung;
```

3. Spalten aus Fahrzeug entfernen

```
ALTER TABLE Fahrzeug
DROP COLUMN Kapazität,
DROP COLUMN Batteriestatus,
DROP COLUMN Ladezeit,
DROP COLUMN CO2_Einsparung;
```

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
```

Fahrzeug_ID	Lieferung_Lieferungsnummer	Wartungsstatus	Fahrzeugtyp_ID
1	1	NULL	1
3	3	NULL	2
5	3	NULL	3
6	3	NULL	4
9	NULL	NULL	1
10	1	NULL	NULL
11	3	NULL	5
12	4	NULL	6
13	NULL	NULL	2
14	NULL	NULL	NULL
15	2	NULL	3
16	5	NULL	4
17	NULL	NULL	NULL
18	NULL	NULL	7

14 rows in set (0,002 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeugtyp;
```

Fahrzeugtyp_ID	Bezeichnung	Kapazität	Batteriestatus	Ladezeit	CO2_Einsparung
1	Elektro-LKW	5000	85	NULL	NULL
2	Elektro-Van	1100	75	NULL	NULL
3	Autonomer LKW	7000	100	NULL	NULL
4	Wasserstoff-LKW	6500	100	NULL	NULL
5	Hybrid-LKW	NULL	80	NULL	NULL
6	Drohne	50	NULL	NULL	NULL
7	Hybrid-Drohne	85	95	NULL	NULL

7 rows in set (0,001 sec)

Anhand des obigen Ergebnisses sieht man die Umstrukturierung der Tabellen in die 2NF.

Im vierten Schritt geht es darum die dritte Normalform 3NF zustande zu bringen. Die Probleme die wir haben sind es, dass die Spalte „Wartungsstatus“ nicht direkt vom Fahrzeugtyp abhängig ist, sondern eher eine Eigenschaft des Fahrzeugs selbst ist. Aus dem Grund muss „Wartungsstatus“ in eine eigene Tabelle ausgelagert werden.

SQL-Schritte zur Umsetzung dieser Maßnahmen:

1. Neue Tabelle „Wartung“ erstellen

```
CREATE TABLE Wartung (
    Fahrzeug_ID INT PRIMARY KEY,
    Wartungsstatus VARCHAR(255),
    FOREIGN KEY (Fahrzeug_ID) REFERENCES Fahrzeug(Fahrzeug_ID)
);
```

2. Wartungsdaten in die neue Tabelle übertragen

```
INSERT INTO Wartung (Fahrzeug_ID, Wartungsstatus)
SELECT Fahrzeug_ID, Wartungsstatus FROM Fahrzeug;
```

3. Spalte „Wartungsstatus“ aus der Fahrzeug-Tabelle entfernen

```
ALTER TABLE Fahrzeug DROP COLUMN Wartungsstatus;
```

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeug;
+-----+-----+-----+
| Fahrzeug_ID | Lieferung_Lieferungsnummer | Fahrzeugtyp_ID |
+-----+-----+-----+
| 1 | 1 | 1 |
| 3 | 3 | 2 |
| 5 | 3 | 3 |
| 6 | 3 | 4 |
| 9 | NULL | 1 |
| 10 | 1 | NULL |
| 11 | 3 | 5 |
| 12 | 4 | 6 |
| 13 | NULL | 2 |
| 14 | NULL | NULL |
| 15 | 2 | 3 |
| 16 | 5 | 4 |
| 17 | NULL | NULL |
| 18 | NULL | 7 |
+-----+-----+-----+
14 rows in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT * FROM Fahrzeugtyp;
+-----+-----+-----+-----+-----+-----+
| Fahrzeugtyp_ID | Bezeichnung | Kapazität | Batteriestatus | Ladezeit | CO2_Einsparung |
+-----+-----+-----+-----+-----+-----+
| 1 | Elektro-LKW | 5000 | 85 | NULL | NULL |
| 2 | Elektro-Van | 1100 | 75 | NULL | NULL |
| 3 | Autonomer LKW | 7000 | 100 | NULL | NULL |
| 4 | Wasserstoff-LKW | 6500 | 100 | NULL | NULL |
| 5 | Hybrid-LKW | NULL | 80 | NULL | NULL |
| 6 | Drohne | 50 | NULL | NULL | NULL |
| 7 | Hybrid-Drohne | 85 | 95 | NULL | NULL |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT * FROM Wartung;
+-----+-----+
| Fahrzeug_ID | Wartungsstatus |
+-----+-----+
| 1 | NULL |
| 3 | NULL |
| 5 | NULL |
| 6 | NULL |
| 9 | NULL |
| 10 | NULL |
| 11 | NULL |
| 12 | NULL |
| 13 | NULL |
| 14 | NULL |
| 15 | NULL |
| 16 | NULL |
| 17 | NULL |
| 18 | NULL |
+-----+-----+
14 rows in set (0,001 sec)
```

Sowie man anhand von dem Endergebnis sehen kann, sind für die 1NF alle Attribute atomar, für die 2NF alle Nicht-Schlüsselattribute direkt von der Fahrzeug-ID abhängig, und für die 3NF keine transitiven Abhängigkeiten mehr vorhanden.

Nun geht es darum die Tabelle Lagerhaus bis zur dritten Normalform zu normalisieren. Dies erfolgt in vier Schritten.

Im ersten Schritt analysieren wir die aktuelle Struktur der Tabelle Lagerhaus:

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
```

Lager_ID	Standort	Kapazität	Energiequelle	Sicherheitsstufe	Kühlkapazität	Drohnenlandeplatz
1	Berlin, Lagerstraße 12	50000	Solar	NULL	NULL	NULL
2	Hamburg, Hafenlager 23	75000	Wind	NULL	NULL	NULL
3	München, Südallee 45	35000	Solar	NULL	NULL	NULL
4	Frankfurt, Westhafen 8	100000	Solar	NULL	NULL	NULL
5	Leipzig, Zentrallager	45000	Erneuerbar	NULL	NULL	NULL
6	Stuttgart, Hauptlager 1	55000	Erneuerbar	NULL	NULL	NULL
7	Düsseldorf, Logistikzentrum	50000	Erneuerbar	NULL	NULL	NULL
8	Dresden, Verteilzentrum	60000	Erneuerbar	NULL	NULL	NULL

8 rows in set (0,001 sec)

Die Problematik hinsichtlich der 1NF ergibt sich aus der Beobachtung, dass es mehrere Eigenschaften in einer einzigen Spalte gibt, die zusammengehören könnten (Standort = Stadt + Straße). Zudem gibt es Attribute mit mehrfachen NULL-Werten, die eine separate Tabelle benötigen (z.B. Kühlkapazität, Drohnenlandeplatz). Zur Lösung dieser Problematik lässt sich zum einen der Standort in zwei Spalten, sprich Stadt und Straße, aufteilen. Zum anderen lassen sich Drohnenlandeplatz und Kühlkapazität in eine separate Tabelle Lagerhaus_Eigenschaft verschieben.

SQL-Schritte zur Umsetzung dieser Maßnahmen sind:

1. Neue Spalten für Stadt und Straße in der Lagerhaus-Tabelle hinzufügen

```
ALTER TABLE Lagerhaus  
ADD COLUMN Stadt VARCHAR(255),  
ADD COLUMN Straße VARCHAR(255);
```

2. Daten in die neuen Spalten übertragen

```
UPDATE Lagerhaus  
SET Stadt = SUBSTRING_INDEX(Standort, ',', 1),  
    Straße = SUBSTRING_INDEX(Standort, ',', -1);
```

3. Spalte Standort aus Lagerhaus entfernen

```
ALTER TABLE Lagerhaus DROP COLUMN Standort;
```

4. Neue Tabelle Lagerhaus_Eigenschaft für Kühlkapazität und Drohnenlandeplatz erstellen

```
CREATE TABLE Lagerhaus_Eigenschaften (  
    Lager_ID INT PRIMARY KEY,  
    Kühlkapazität INT,  
    Drohnenlandeplatz BOOLEAN,  
    FOREIGN KEY (Lager_ID) REFERENCES Lagerhaus(Lager_ID)  
)
```

5. Daten in die neue Tabelle übertragen

```
INSERT INTO Lagerhaus_Eigenschaften (Lager_ID, Kühlkapazität, Drohnenlandeplatz)  
SELECT Lager_ID, Kühlkapazität, Drohnenlandeplatz FROM Lagerhaus;
```

6. Spalten aus der Lagerhaus-Tabelle entfernen

```
ALTER TABLE Lagerhaus  
DROP COLUMN Kühlkapazität,  
DROP COLUMN Drohnenlandeplatz;
```

MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;

Lager_ID	Kapazität	Energiequelle	Sicherheitsstufe	Stadt	Straße
1	50000	Solar	NULL	Berlin	Lagerstraße 12
2	75000	Wind	NULL	Hamburg	Hafenlager 23
3	35000	Solar	NULL	München	Südallee 45
4	100000	Solar	NULL	Frankfurt	Westhafen 8
5	45000	Erneuerbar	NULL	Leipzig	Zentrallager
6	55000	Erneuerbar	NULL	Stuttgart	Hauptlager 1
7	50000	Erneuerbar	NULL	Düsseldorf	Logistikzentrum
8	60000	Erneuerbar	NULL	Dresden	Verteilzentrum

8 rows in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus_Eigenschaften;

Lager_ID	Kühlkapazität	Drohnenlandeplatz
1	NULL	NULL
2	NULL	NULL
3	NULL	NULL
4	NULL	NULL
5	NULL	NULL
6	NULL	NULL
7	NULL	NULL
8	NULL	NULL

8 rows in set (0,001 sec)

In Schritt drei geht es darum die Umstrukturierung in die 2NF durchzuführen. Für den Anfang ist zu beobachten, dass die Spalte Energiequelle nur vom Standort abhängig ist, nicht von dem Lager_ID selbst. Des Weiteren ist Sicherheitsstufe ebenfalls eine allgemeine Eigenschaft von Lagerhäusern, nicht spezifisch für jede Lager-ID. Zur Lösung dieser Problematik sollte zum einen eine neue Tabelle Energiequelle, und zum anderen eine neue Tabelle Sicherheitsstufe erstellt werden.

Zur Umsetzung dieser Maßnahmen werden folge SQL-Befehle vorgeführt:

1. Neue Tabelle Energiequelle erstellen

```
CREATE TABLE Energiequelle (
    Energiequelle_ID INT PRIMARY KEY AUTO_INCREMENT,
    Bezeichnung VARCHAR(255) NOT NULL
);
```

2. Bestehende Energiequellen in die neue Tabelle übertragen

```
INSERT INTO Energiequelle (Bezeichnung)
SELECT DISTINCT Energiequelle FROM Lagerhaus WHERE Energiequelle IS NOT NULL;
```

3. Neue Spalte Energiequelle_ID in Lagerhaus hinzufügen

```
ALTER TABLE Lagerhaus
ADD COLUMN Energiequelle_ID INT,
ADD FOREIGN KEY (Energiequelle_ID) REFERENCES Energiequelle(Energiequelle_ID);
```

4. Werte von Energiequelle in die neue ID-Spalte umwandeln

```
UPDATE Lagerhaus l
JOIN Energiequelle e ON l.Energiequelle = e.Bezeichnung
SET l.Energiequelle_ID = e.Energiequelle_ID;
```

5. Spalte Energiequelle aus Lagerhaus entfernen

```
ALTER TABLE Lagerhaus DROP COLUMN Energiequelle;
```

6. Neue Tabelle Sicherheitsstufe erstellen

```
CREATE TABLE Sicherheitsstufe (
    Sicherheitsstufe_ID INT PRIMARY KEY AUTO_INCREMENT,
    Sicherheitsniveau VARCHAR(50)
);
```

7. Sicherheitsstufen in die neue Tabelle übertragen

```
INSERT INTO Sicherheitsstufe (Sicherheitsniveau)
```

```
SELECT DISTINCT Sicherheitsstufe FROM Lagerhaus WHERE Sicherheitsstufe IS NOT NULL;
```

8. Neue Spalte Sicherheitsstufe_ID in Lagerhaus hinzufügen

```
ALTER TABLE Lagerhaus
```

```
ADD COLUMN Sicherheitsstufe_ID INT,
```

```
ADD FOREIGN KEY (Sicherheitsstufe_ID) REFERENCES Sicherheitsstufe(Sicherheitsstufe_ID);
```

9. Werte in die neue Sicherheitsstufen-Spalte übertragen

```
UPDATE Lagerhaus I
```

```
JOIN Sicherheitsstufe s ON I.Sicherheitsstufe = s.Sicherheitsniveau
```

```
SET I.Sicherheitsstufe_ID = s.Sicherheitsstufe_ID;
```

10. Spalte Sicherheitsstufe aus Lagerhaus entfernen

```
ALTER TABLE Lagerhaus DROP COLUMN Sicherheitsstufe;
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
```

Lager_ID	Kapazität	Stadt	Straße	Energiequelle_ID	Sicherheitsstufe_ID
1	50000	Berlin	Lagerstraße 12	1	NULL
2	75000	Hamburg	Hafenlager 23	2	NULL
3	35000	München	Südallee 45	1	NULL
4	100000	Frankfurt	Westhafen 8	1	NULL
5	45000	Leipzig	Zentrallager	3	NULL
6	55000	Stuttgart	Hauptlager 1	3	NULL
7	50000	Düsseldorf	Logistikzentrum	3	NULL
8	60000	Dresden	Verteilzentrum	3	NULL

```
8 rows in set (0,002 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus_Eigenschaften;
```

Lager_ID	Kühlkapazität	Drohnenlandeplatz
1	NULL	NULL
2	NULL	NULL
3	NULL	NULL
4	NULL	NULL
5	NULL	NULL
6	NULL	NULL
7	NULL	NULL
8	NULL	NULL

```
8 rows in set (0,006 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Energiequelle;
```

Energiequelle_ID	Bezeichnung
1	Solar
2	Wind
3	Erneuerbar

```
3 rows in set (0,001 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Sicherheitsstufe;
```

```
Empty set (0,001 sec)
```

So sieht das Ergebnis aus. Die Tabelle Sicherheitsstufe ist leer, da es hierfür keine Einträge gegeben hat. Auch vorher nicht.

Nun gehen wir zu Schritt vier rüber, um die Umstrukturierung in die 3NF durchzuführen. Die Problematik die sich ergibt ist der, dass die Kapazität eine Eigenschaft des Lagerhauses ist und nicht von der Lagerhaus-ID direkt abhängig ist. Aus dem Grund sollte sie in eine separate Tabelle ausgelagert werden. Zur Lösung dieser sollte eine neue Tabelle Lagerkapazität für Kapazitätswerte erstellt werden.

Zur Umsetzung dieser Schritte sind folgende SQL-Befehle anzuwenden:

1. Neue Tabelle Lagerkapazität erstellen

```
CREATE TABLE Lagerkapazität (
    Lagerkapazität_ID INT PRIMARY KEY AUTO_INCREMENT,
    Kapazität INT NOT NULL
);
```

2. Bestehende Kapazitätswerte in die neue Tabelle übertragen

```
INSERT INTO Lagerkapazität (Kapazität)
SELECT DISTINCT Kapazität FROM Lagerhaus WHERE Kapazität IS NOT NULL;
```

3. Neue Spalte Lagerkapazität_ID in Lagerhaus hinzufügen

```
ALTER TABLE Lagerhaus
ADD COLUMN Lagerkapazität_ID INT,
ADD FOREIGN KEY (Lagerkapazität_ID) REFERENCES Lagerkapazität(Lagerkapazität_ID);
```

4. Werte in die neue Lagerkapazitäts-Spalte übertragen

```
UPDATE Lagerhaus l
JOIN Lagerkapazität lk ON l.Kapazität = lk.Kapazität
SET l.Lagerkapazität_ID = lk.Lagerkapazität_ID;
```

5. Spalte Kapazität aus Lagerhaus entfernen

```
ALTER TABLE Lagerhaus DROP COLUMN Kapazität;
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
```

Lager_ID	Stadt	Straße	Energiequelle_ID	Sicherheitsstufe_ID	Lagerkapazität_ID
1	Berlin	Lagerstraße 12	1	NULL	1
2	Hamburg	Hafenlager 23	2	NULL	2
3	München	Südallee 45	1	NULL	3
4	Frankfurt	Westhafen 8	1	NULL	4
5	Leipzig	Zentrallager	3	NULL	5
6	Stuttgart	Hauptlager 1	3	NULL	6
7	Düsseldorf	Logistikzentrum	3	NULL	1
8	Dresden	Verteilzentrum	3	NULL	7

8 rows in set (0,002 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus_Eigenschaft;
```

```
ERROR 1146 (42S02): Table 'hosakbari_db.Lagerhaus_Eigenschaft' doesn't exist
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus;
```

Lager_ID	Stadt	Straße	Energiequelle_ID	Sicherheitsstufe_ID	Lagerkapazität_ID
1	Berlin	Lagerstraße 12	1	NULL	1
2	Hamburg	Hafenlager 23	2	NULL	2
3	München	Südallee 45	1	NULL	3
4	Frankfurt	Westhafen 8	1	NULL	4
5	Leipzig	Zentrallager	3	NULL	5
6	Stuttgart	Hauptlager 1	3	NULL	6
7	Düsseldorf	Logistikzentrum	3	NULL	1
8	Dresden	Verteilzentrum	3	NULL	7

8 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerhaus_Eigenschaften;
```

Lager_ID	Kühlkapazität	Drohnenlandeplatz
1	NULL	NULL
2	NULL	NULL
3	NULL	NULL
4	NULL	NULL
5	NULL	NULL
6	NULL	NULL
7	NULL	NULL
8	NULL	NULL

8 rows in set (0,002 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Energiequelle;
```

Energiequelle_ID	Bezeichnung
1	Solar
2	Wind
3	Erneuerbar

3 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Sicherheitsstufe;
```

Empty set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Lagerkapazität;
```

Lagerkapazität_ID	Kapazität
1	50000
2	75000
3	35000
4	100000
5	45000
6	55000
7	60000

7 rows in set (0,001 sec)

Das folgende Ergebnis zeigt das Resultat der Umstrukturierung von Lagerhaus bis dritte Normalform.

Nun werden wir die Tabelle Lieferung bis zur dritten Normalform normalisieren. Dies erfolgt in mehreren Schritten. Als aller erst werden wir die Tabelle Lieferung analysieren.

Lieferungsnummer	Zielort	Lieferzeitpunkt	Status	Priorität	Transporttyp	Versandkosten
1	Berlin, Alexanderplatz	2044-06-15	unterwegs	NULL	NULL	NULL
2	Hamburg, Hafenstraße	2044-06-16	gelfert	NULL	NULL	NULL
3	München, Marienplatz	2044-06-17	verspätet	NULL	NULL	NULL
4	Frankfurt, Zeilstraße	2044-06-18	geplant	NULL	NULL	NULL
5	Unbekannt	2044-06-20	in Bearbeitung	NULL	NULL	NULL
6	Unbekannt	2044-06-21	Verspätet	NULL	NULL	NULL
7	Unbekannt	2044-06-22	Verspätet	NULL	NULL	NULL
8	Unbekannt	2044-06-23	Verspätet	NULL	NULL	NULL
9	Köln, Domplatz	2044-06-25	geplant	2	Drohne	19.99
10	NULL	2044-06-27	unterwegs	1	Elektro-LKW	29.99
11	Hamburg, Reeperbahn	NULL	verspätet	3	Hybrid-LKW	24.50
12	München, Stachus	2044-07-01	gelfert	2	Elektro-Van	15.75
13	NULL	NULL	in Bearbeitung	1	Drohne	12.99
14	Düsseldorf, Altstadt	2044-07-05	NULL	2	Wasserstoff-LKW	22.30
15	Stuttgart, Schlossplatz	2044-06-29	unterwegs	1	Elektro-Scooter	9.99
16	Frankfurt, Mainufer	NULL	NULL	3	Drohne	14.50
17	NULL	2044-07-03	geplant	2	Elektro-LKW	17.89
18	Berlin, Kudamm	NULL	in Bearbeitung	1	Hybrid-Drohne	11.99
19	NULL	NULL	NULL	3	Autonomer LKW	31.50
20	Nürnberg, Zentrum	2044-07-10	verspätet	1	Drohne	10.99
21	Bremen, Innenstadt	2044-07-12	NULL	NULL	NULL	NULL
22	NULL	2044-07-13	geplant	NULL	NULL	NULL
23	Hannover, Hauptbahnhof	NULL	verspätet	NULL	NULL	NULL
24	NULL	NULL	NULL	NULL	NULL	NULL

24 rows in set (0,001 sec)

Im ersten Schritt wird es darum gehen die Tabelle in die erste Normalform umzustrukturieren. Die Problematik die sich hierzu ergibt ist der, dass zum ersten der Zielort zwei Daten enthält (Stadt + Straße) in einer Spalte, zum zweiten, dass die Spalte Status wiederkehrende Werte enthält (z.B. unterwegs, geplant, verspätet). Zum dritten ist Transporttyp direkt in der Tabelle gespeichert, anstatt mit einer separaten Tabelle referenziert zu werden. Zur Lösung hiervon müssen zum ersten Zielort in Stadt und Straße aufgeteilt werden, zum zweiten eine separate Tabelle für Lieferstatus erstellt werden und zum dritten für Transporttyp eine separate Tabelle erstellt werden.

Die SQL-Schritte hierzu sind:

1. Neue Spalten Stadt und Straße in der Tabelle Lieferung hinzufügen

```
ALTER TABLE Lieferung
ADD COLUMN Stadt VARCHAR(255),
ADD COLUMN Straße VARCHAR(255);
```

2. Daten übertragen (falls Zielort nicht NUL ist)

```
UPDATE Lieferung
SET Stadt = SUBSTRING_INDEX(Zielort, ',', 1),
    Straße = SUBSTRING_INDEX(Zielort, ',', -1);
```

3. Spalte Zielort entfernen

```
ALTER TABLE Lieferung DROP COLUMN Zielort;
```

4. Neue Tabelle Lieferstatus erstellen

```
CREATE TABLE Lieferstatus (
    Status_ID INT PRIMARY KEY AUTO_INCREMENT,
    Bezeichnung VARCHAR(255) NOT NULL
);
```

5. Bestehende Statuswerte in die neue Tabelle übertragen

```
INSERT INTO Lieferstatus (Bezeichnung)
SELECT DISTINCT Status FROM Lieferung WHERE Status IS NOT NULL;
```

6. Neue Spalte Status_ID in Lieferung hinzufügen

```
ALTER TABLE Lieferung
ADD COLUMN Status_ID INT,
ADD FOREIGN KEY (Status_ID) REFERENCES Lieferstatus(Status_ID);
```

7. Werte von Status in die neue Status-ID-Spalte umwandeln

```
UPDATE Lieferung l
```

```
JOIN Lieferstatus ls ON l.Status = ls.Bezeichnung
SET l.Status_ID = ls.Status_ID;
8. Spalte Status aus Lieferung entfernen
ALTER TABLE Lieferung DROP COLUMN Status;
9. Neue Tabelle Transporttyp erstellen
CREATE TABLE Transporttyp (
    Transporttyp_ID INT PRIMARY KEY AUTO_INCREMENT,
    Bezeichnung VARCHAR(255) NOT NULL
);
10. Bestehende Transporttypen in die neue Tabelle übertragen
INSERT INTO Transporttyp (Bezeichnung)
SELECT DISTINCT Transporttyp FROM Lieferung WHERE Transporttyp IS NOT NULL;
11. Neue Spalte Transporttyp_ID in Lieferung hinzufügen
ALTER TABLE Lieferung
ADD COLUMN Transporttyp_ID INT,
ADD FOREIGN KEY (Transporttyp_ID) REFERENCES Transporttyp(Transporttyp_ID);
12. Werte von Transporttyp in die neue ID-Spalte umwandeln
UPDATE Lieferung l
JOIN Transporttyp t ON l.Transporttyp = t.Bezeichnung
SET l.Transporttyp_ID = t.Transporttyp_ID;
13. Spalte Transporttyp aus Lieferung entfernen
ALTER TABLE Lieferung DROP COLUMN Transporttyp;
```

Lieferungsnummer	Lieferzeitpunkt	Priorität	Versandkosten	Stadt	Straße	Status_ID	Transporttyp_ID
1	2044-06-15	NULL	NULL	Berlin	Alexanderplatz	1	NULL
2	2044-06-16	NULL	NULL	Hamburg	Hafenstraße	2	NULL
3	2044-06-17	NULL	NULL	München	Marienplatz	3	NULL
4	2044-06-18	NULL	NULL	Frankfurt	Zeilstraße	4	NULL
5	2044-06-20	NULL	NULL	Unbekannt	Unbekannt	5	NULL
6	2044-06-21	NULL	NULL	Unbekannt	Unbekannt	3	NULL
7	2044-06-22	NULL	NULL	Unbekannt	Unbekannt	3	NULL
8	2044-06-23	NULL	NULL	Unbekannt	Unbekannt	3	NULL
9	2044-06-25	2	19.99	Köln	Domplatz	4	1
10	2044-06-27	1	29.99	NULL	NULL	1	2
11	NULL	3	24.50	Hamburg	Reeperbahn	3	3
12	2044-07-01	2	15.75	München	Stachus	2	4
13	NULL	1	12.99	NULL	NULL	5	1
14	2044-07-05	2	22.30	Düsseldorf	Altstadt	NULL	5
15	2044-06-29	1	9.99	Stuttgart	Schlossplatz	1	6
16	NULL	3	14.50	Frankfurt	Mainufer	NULL	1
17	2044-07-03	2	17.89	NULL	NULL	4	2
18	NULL	1	11.99	Berlin	Kudamm	5	7
19	NULL	3	31.50	NULL	NULL	NULL	8
20	2044-07-10	1	10.99	Nürnberg	Zentrum	3	1
21	2044-07-12	NULL	NULL	Bremen	Innenstadt	NULL	NULL
22	2044-07-13	NULL	NULL	NULL	NULL	4	NULL
23	NULL	NULL	NULL	Hannover	Hauptbahnhof	3	NULL
24	NULL	NULL	NULL	NULL	NULL	NULL	NULL

24 rows in set (0,002 sec)

MariaDB [hosakbari_db]> SELECT * FROM Lieferstatus;

Status_ID	Bezeichnung
1	unterwegs
2	geliefert
3	verspätet
4	geplant
5	in Bearbeitung

5 rows in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT * FROM Transporttyp;

Transporttyp_ID	Bezeichnung
1	Drohne
2	Elektro-LKW
3	Hybrid-LKW
4	Elektro-Van
5	Wasserstoff-LKW
6	Elektro-Scooter
7	Hybrid-Drohne
8	Autonomer LKW

8 rows in set (0,001 sec)

Im zweiten Schritt geht es darum die Umstrukturierung in die 2NF vorzunehmen. Die Problematik indem Sinne ist es, das zu ersten die Versandkosten nicht direkt von der Lieferungsnummer abhängen, sondern vom Transporttyp. Zum zweiten könnte Priorität in eine separate Tabelle verschoben, da sie eine begrenzte Anzahl von Werten hat. Zur Lösung hierzu muss eine neue Tabelle Versandkosten für Transportkosten erstellt werden. Zum zweiten muss eine neue Tabelle Lieferpriorität für verschiedene Prioritäten erstellt werden.

SQL-Schritte zur Umsetzung hiervon umfassen:

1. Neue Tabelle Versandkosten erstellen

```
CREATE TABLE Versandkosten (
    Transporttyp_ID INT PRIMARY KEY,
    Kosten DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (Transporttyp_ID) REFERENCES Transporttyp(Transporttyp_ID)
);
```

2. Bestehende Versandkosten in die neue Tabelle übertragen

```
INSERT INTO Versandkosten (Transporttyp_ID, Kosten)
SELECT DISTINCT Transporttyp_ID, Versandkosten FROM Lieferung WHERE Versandkosten IS NOT NULL;
```

```
MariaDB [hosakbari_db]> INSERT INTO Versandkosten (Transporttyp_ID, Kosten)
en FROM Lieferun → SELECT DISTINCT Transporttyp_ID, Versandkosten FROM Lieferung WHERE Versandkosten IS NOT NULL;
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
MariaDB [hosakbari_db]>
```

Wir stoßen hierbei zu einem Problem: ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'

Diese Fehlermeldung gibt an, dass beim Einfügen in die Tabelle Versandkosten ein doppelter Primärschlüsselwert für Transporttyp aufgetreten ist. Das passiert aus dem Grund, weil in der Tabelle Versandkosten Transporttyp_ID der Primärschlüssel ist. Beim INSERT INTO Versandkosten wurde versucht, doppelte Werte für Transporttyp_ID einzufügen. Die Ursache hierzu ist es, weil es in der Tabelle Lieferung mehrere Zeilen mit dem gleichen Transporttyp_ID gibt, aber mit unterschiedlichen Versandkosten.

Durch die folgende Abfrage wird wiedergegeben, dass es möglicherweise mehrere verschiedene Versandkosten für denselben Transporttyp_ID gibt:

```
SELECT DISTINCT Transporttyp_ID, Versandkosten FROM Lieferung WHERE Versandkosten IS NOT NULL;
```

```
MariaDB [hosakbari_db]> SELECT DISTINCT Transporttyp_ID, Versandkosten FROM Lieferung WHERE Versandkosten IS NOT NULL;
+-----+-----+
| Transporttyp_ID | Versandkosten |
+-----+-----+
| 1 | 19.99 |
| 2 | 29.99 |
| 3 | 24.50 |
| 4 | 15.75 |
| 1 | 12.99 |
| 5 | 22.30 |
| 6 | 9.99 |
| 1 | 14.50 |
| 2 | 17.89 |
| 7 | 11.99 |
| 8 | 31.50 |
| 1 | 10.99 |
+-----+-----+
12 rows in set (0,001 sec)
```

Wie man es hier sehen kann, gibt es mehrere unterschiedliche Versandkosten für die gleiche Transporttyp_ID. Da Transporttyp_ID der Primärschlüssel ist, darf es nur einmal pro Transporttyp existieren. Aus dem Grund kommt es zu einer Fehlermeldung.

Man kann mit dem folgenden Befehl überprüfen, ob es mehrere verschiedene Versandkosten für einen Transporttyp gibt:

```
SELECT Transporttyp_ID, COUNT(DISTINCT Versandkosten) AS Anzahl_Versandkosten
FROM Lieferung
WHERE Versandkosten IS NOT NULL
GROUP BY Transporttyp_ID
HAVING COUNT(DISTINCT Versandkosten) > 1;
```

```
MariaDB [hosakbari_db]> SELECT Transporttyp_ID, COUNT(DISTINCT Versandkosten) AS Anzahl_Versandkosten
→ FROM Lieferung
→ WHERE Versandkosten IS NOT NULL
→ GROUP BY Transporttyp_ID
→ HAVING COUNT(DISTINCT Versandkosten) > 1;
+-----+-----+
| Transporttyp_ID | Anzahl_Versandkosten |
+-----+-----+
| 1 | 4 |
| 2 | 2 |
+-----+-----+
2 rows in set (0,002 sec)
```

Hierbei kann man sehen, dass es für Transporttyp_ID 1, 4 verschiedene Versandkosten gibt. Für ID 2 gibt es 2 verschiedene Versandkosten.

Wir nehmen hierzu eine kleine Änderung vor und passen die Tabelle Versandkosten so an, dass unterschiedliche Versandkosten je Transporttyp erlaubt sind. Pro Transporttyp sollen mehrere unterschiedliche Versandkosten möglich sein, wofür wir die Struktur der Tabelle Versandkosten anpassen müssen.

Aktuell ist es so, dass in der Tabelle Transporttyp, Transporttyp_ID Primärschlüssel dieser Tabelle ist. Das bleibt unverändert. In Tabelle Lieferung ist Transporttyp_ID Fremdschlüssel. Falls hier Transporttyp_ID entfernt oder geändert wird, könnte dies zu Fremdschlüsselverletzungen führen. In der Tabelle Versandkosten ist Transporttyp_ID aktuell der Fremdschlüssel. Wir wollen das so haben, dass ein Transporttyp mehrere verschiedene Versandkosten haben kann. Hierzu muss es eine weitere Spalte in den Primärschlüssel aufgenommen werden.

Da unterschiedliche Versandkosten pro Transporttyp_ID möglich sein sollen, wird Lieferungsnummer als zusätzliche Primärschlüsselspalte aufgenommen:

1. Bestehenden Primärschlüssel entfernen: Falls bereits Fremdschlüssel vorhanden sind, müssen diese zuerst entfernt werden:

```
ALTER TABLE Versandkosten DROP FOREIGN KEY Versandkosten_ibfk_1;  
ALTER TABLE Versandkosten DROP PRIMARY KEY;
```

2. Neuen Primärschlüssel setzen: Nun kann Lieferungsnummer in den Primärschlüssel aufgenommen werden:

```
ALTER TABLE Versandkosten ADD COLUMN Lieferungsnummer INT NOT NULL;  
ALTER TABLE Versandkosten ADD PRIMARY KEY (Transporttyp_ID, Lieferungsnummer);
```

Nun ist jede Kombination aus Transporttyp_ID und Lieferungsnummer einzigartig. Damit kann es mehrere Versandkosten pro Transporttyp geben, die jeweils einer Lieferung zugeordnet sind.

3. Fremdschlüsselbeziehung zur Lieferung wiederherstellen: Da Lieferungsnummer jetzt als Teil des Primärschlüssels ist, muss ein neuer Fremdschlüssel zur Lieferung gesetzt werden

```
ALTER TABLE Versandkosten ADD FOREIGN KEY (Lieferungsnummer) REFERENCES Lieferung(Lieferungsnummer);
```

Als letzten Schritt müssen wir aus der Tabelle Lieferung die Versandkosten in die Tabelle Versandkosten korrekt (mit neuer Anpassung der Versandkosten-Tabelle) hinzufügen:

```
INSERT INTO Versandkosten (Transporttyp_ID, Lieferungsnummer, Kosten)  
SELECT DISTINCT Transporttyp_ID, Lieferungsnummer, Versandkosten  
FROM Lieferung  
WHERE Versandkosten IS NOT NULL;
```

Lieferungsnummer	Lieferzeitpunkt	Priorität	Versandkosten	Stadt	Straße	Status_ID	Transporttyp_ID
1	2044-06-15	NULL	NULL	Berlin	Alexanderplatz	1	NULL
2	2044-06-16	NULL	NULL	Hamburg	Hafenstraße	2	NULL
3	2044-06-17	NULL	NULL	München	Marienplatz	3	NULL
4	2044-06-18	NULL	NULL	Frankfurt	Zeilstraße	4	NULL
5	2044-06-20	NULL	NULL	Unbekannt	Unbekannt	5	NULL
6	2044-06-21	NULL	NULL	Unbekannt	Unbekannt	3	NULL
7	2044-06-22	NULL	NULL	Unbekannt	Unbekannt	3	NULL
8	2044-06-23	NULL	NULL	Unbekannt	Unbekannt	3	NULL
9	2044-06-25	2	19.99	Köln	Domplatz	4	1
10	2044-06-27	1	29.99	NULL	NULL	1	2
11	NULL	3	24.50	Hamburg	Reeperbahn	3	3
12	2044-07-01	2	15.75	München	Stachus	2	4
13	NULL	1	12.99	NULL	NULL	5	1
14	2044-07-05	2	22.30	Düsseldorf	Altstadt	NULL	5
15	2044-06-29	1	9.99	Stuttgart	Schlossplatz	1	6
16	NULL	3	14.50	Frankfurt	Mainufer	NULL	1
17	2044-07-03	2	17.89	NULL	NULL	4	2
18	NULL	1	11.99	Berlin	Kudamm	5	7
19	NULL	3	31.50	NULL	NULL	NULL	8
20	2044-07-10	1	10.99	Nürnberg	Zentrum	3	1
21	2044-07-12	NULL	NULL	Bremen	Innenstadt	NULL	NULL
22	2044-07-13	NULL	NULL	NULL	NULL	4	NULL
23	NULL	NULL	NULL	Hannover	Hauptbahnhof	3	NULL
24	NULL	NULL	NULL	NULL	NULL	NULL	NULL

24 rows in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT * FROM Transporttyp;

Transporttyp_ID	Bezeichnung
1	Drohne
2	Elektro-LKW
3	Hybrid-LKW
4	Elektro-Van
5	Wasserstoff-LKW
6	Elektro-Scooter
7	Hybrid-Drohne
8	Autonomer LKW

8 rows in set (0,001 sec)

MariaDB [hosakbari_db]> SELECT * FROM Versandkosten;

Transporttyp_ID	Kosten	Lieferungsnummer
1	19.99	9
1	12.99	13
1	14.50	16
1	10.99	20
2	29.99	10
2	17.89	17
3	24.50	11
4	15.75	12
5	22.30	14
6	9.99	15
7	11.99	18
8	31.50	19

12 rows in set (0,001 sec)

Sowie man sehen kann, sind nun mehrere Einträge für denselben Transporttyp_ID mit unterschiedlichen Lieferungsnummern gegeben.

Wir hatten ursprünglich während der zweiten Normalform die Problematik gehabt: ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'

Dieses Problem haben wir jetzt jedoch gelöst. Nun wollen wir mit der 2NF weitermachen: Da wir die Versandkosten in einer eigenen Tabelle gespeichert haben, kann die Spalte Versandkosten aus der Tabelle Lieferung entfernt werden:

ALTER TABLE Lieferung DROP COLUMN Versandkosten;

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
```

Lieferungsnummer	Lieferzeitpunkt	Priorität	Stadt	Straße	Status_ID	Transporttyp_ID
1	2044-06-15	NULL	Berlin	Alexanderplatz	1	NULL
2	2044-06-16	NULL	Hamburg	Hafenstraße	2	NULL
3	2044-06-17	NULL	München	Marienplatz	3	NULL
4	2044-06-18	NULL	Frankfurt	Zeilstraße	4	NULL
5	2044-06-20	NULL	Unbekannt	Unbekannt	5	NULL
6	2044-06-21	NULL	Unbekannt	Unbekannt	3	NULL
7	2044-06-22	NULL	Unbekannt	Unbekannt	3	NULL
8	2044-06-23	NULL	Unbekannt	Unbekannt	3	NULL
9	2044-06-25	2	Köln	Domplatz	4	1
10	2044-06-27	1	NULL	NULL	1	2
11	NULL	3	Hamburg	Reeperbahn	3	3
12	2044-07-01	2	München	Stachus	2	4
13	NULL	1	NULL	NULL	5	1
14	2044-07-05	2	Düsseldorf	Altstadt	NULL	5
15	2044-06-29	1	Stuttgart	Schlossplatz	1	6
16	NULL	3	Frankfurt	Mainufer	NULL	1
17	2044-07-03	2	NULL	NULL	4	2
18	NULL	1	Berlin	Kudamm	5	7
19	NULL	3	NULL	NULL	NULL	8
20	2044-07-10	1	Nürnberg	Zentrum	3	1
21	2044-07-12	NULL	Bremen	Innenstadt	NULL	NULL
22	2044-07-13	NULL	NULL	NULL	4	NULL
23	NULL	NULL	Hannover	Hauptbahnhof	3	NULL
24	NULL	NULL	NULL	NULL	NULL	NULL

24 rows in set (0,002 sec)

Nun müssen wir analysieren, ob noch weitere funktionale Abhängigkeiten in der Tabelle Lieferung bestehen, die nicht direkt von der Primärschlüssel Spalte Lieferungsnummer abhängen. Wir stellen fest:

Die Werte für Priorität sind wiederkehrend (z.B. NULL, 1, 2, 3). Das bedeutet, dass sie in eine eigene Tabelle ausgelagert werden sollten. Des Weiteren können mehrere Lieferungen an die gleiche Adresse gehen. Das führt zu Datenredundanz. Die Adresse sollte deshalb in eine eigene Tabelle ausgelagert werden. Zur Lösung hierzu sollte eine neue Tabelle Lieferprioritäten erstellt werden, sowie eine neue Tabelle Lieferadressen.

Die SQL-Schritte zur Umsetzung dieser Maßnahmen sind:

1. Neue Tabelle Lieferpriorität erstellen

```
CREATE TABLE Lieferpriorität (
    Priorität_ID INT PRIMARY KEY AUTO_INCREMENT,
    Bezeichnung VARCHAR(50)
);
```

2. Bestehende Prioritätswerte in die neue Tabelle übertragen

```
INSERT INTO Lieferpriorität (Bezeichnung)
SELECT DISTINCT Priorität FROM Lieferung WHERE Priorität IS NOT NULL;
```

3. Neue Spalte Priorität_ID in Lieferung hinzufügen

```
ALTER TABLE Lieferung
ADD COLUMN Priorität_ID INT,
ADD FOREIGN KEY (Priorität_ID) REFERENCES Lieferpriorität(Priorität_ID);
```

4. Werte von Priorität in die neue ID-Spalte umwandeln

```
UPDATE Lieferung l
JOIN Lieferpriorität p ON l.Priorität = p.Bezeichnung
SET l.Priorität_ID = p.Priorität_ID;
```

5. Spalte Priorität aus Lieferung entfernen

```
ALTER TABLE Lieferung DROP COLUMN Priorität;
```

MariaDB [hosakbari_db]> SELECT * FROM Lieferung;

Lieferungsnummer	Lieferzeitpunkt	Stadt	Straße	Status_ID	Transporttyp_ID	Priorität_ID
1	2044-06-15	Berlin	Alexanderplatz	1	NULL	NULL
2	2044-06-16	Hamburg	Hafenstraße	2	NULL	NULL
3	2044-06-17	München	Marienplatz	3	NULL	NULL
4	2044-06-18	Frankfurt	Zeilstraße	4	NULL	NULL
5	2044-06-20	Unbekannt	Unbekannt	5	NULL	NULL
6	2044-06-21	Unbekannt	Unbekannt	3	NULL	NULL
7	2044-06-22	Unbekannt	Unbekannt	3	NULL	NULL
8	2044-06-23	Unbekannt	Unbekannt	3	NULL	NULL
9	2044-06-25	Köln	Domplatz	4	1	1
10	2044-06-27	NULL	NULL	1	2	2
11	NULL	Hamburg	Reeperbahn	3	3	3
12	2044-07-01	München	Stachus	2	4	1
13	NULL	NULL	NULL	5	1	2
14	2044-07-05	Düsseldorf	Altstadt	NULL	5	1
15	2044-06-29	Stuttgart	Schlossplatz	1	6	2
16	NULL	Frankfurt	Mainufer	NULL	1	3
17	2044-07-03	NULL	NULL	4	2	1
18	NULL	Berlin	Kudamm	5	7	2
19	NULL	NULL	NULL	NULL	8	3
20	2044-07-10	Nürnberg	Zentrum	3	1	2
21	2044-07-12	Bremen	Innenstadt	NULL	NULL	NULL
22	2044-07-13	NULL	NULL	4	NULL	NULL
23	NULL	Hannover	Hauptbahnhof	3	NULL	NULL
24	NULL	NULL	NULL	NULL	NULL	NULL

24 rows in set (0,002 sec)

MariaDB [hosakbari_db]> SELECT * FROM Lieferpriorität;

Priorität_ID	Bezeichnung
1	2
2	1
3	3

3 rows in set (0,001 sec)

Nun setzen wir die Maßnahmen für die Tabelle Lieferadresse um. Die SQL-Schritte hierzu umfassen:

1. Neue Tabelle Lieferadresse erstellen

```
CREATE TABLE Lieferadresse (
    Adresse_ID INT PRIMARY KEY AUTO_INCREMENT,
    Stadt VARCHAR(255),
    Straße VARCHAR(255)
);
```

2. Bestehende Adressen in die neue Tabelle übertragen

```
INSERT INTO Lieferadresse (Stadt, Straße)
SELECT DISTINCT Stadt, Straße FROM Lieferung WHERE Stadt IS NOT NULL AND Straße IS NOT NULL;
```

3. Neue Spalte Adresse_ID in Lieferung hinzufügen

```
ALTER TABLE Lieferung
ADD COLUMN Adresse_ID INT,
ADD FOREIGN KEY (Adresse_ID) REFERENCES Lieferadresse(Adresse_ID);
```

4. Werte von Stadt und Straße in die neue Tabelle umwandeln

```
UPDATE Lieferung l
JOIN Lieferadresse a ON l.Stadt = a.Stadt AND l.Straße = a.Straße
SET l.Adresse_ID = a.Adresse_ID;
```

5. Spalten Stadt und Straße aus Lieferung entfernen

```
ALTER TABLE Lieferung DROP COLUMN Stadt, DROP COLUMN Straße;
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
```

Lieferungsnummer	Lieferzeitpunkt	Status_ID	Transporttyp_ID	Priorität_ID	Adresse_ID
1	2044-06-15	1	NULL	NULL	1
2	2044-06-16	2	NULL	NULL	2
3	2044-06-17	3	NULL	NULL	3
4	2044-06-18	4	NULL	NULL	4
5	2044-06-20	5	NULL	NULL	5
6	2044-06-21	3	NULL	NULL	5
7	2044-06-22	3	NULL	NULL	5
8	2044-06-23	3	NULL	NULL	5
9	2044-06-25	4	1	1	6
10	2044-06-27	1	2	2	NULL
11	NULL	3	3	3	7
12	2044-07-01	2	4	1	8
13	NULL	5	1	2	NULL
14	2044-07-05	NULL	5	1	9
15	2044-06-29	1	6	2	10
16	NULL	NULL	1	3	11
17	2044-07-03	4	2	1	NULL
18	NULL	5	7	2	12
19	NULL	NULL	8	3	NULL
20	2044-07-10	3	1	2	13
21	2044-07-12	NULL	NULL	NULL	14
22	2044-07-13	4	NULL	NULL	NULL
23	NULL	3	NULL	NULL	15
24	NULL	NULL	NULL	NULL	NULL

```
24 rows in set (0,002 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferadresse;
```

Adresse_ID	Stadt	Straße
1	Berlin	Alexanderplatz
2	Hamburg	Hafenstraße
3	München	Marienplatz
4	Frankfurt	Zeilstraße
5	Unbekannt	Unbekannt
6	Köln	Domplatz
7	Hamburg	Reeperbahn
8	München	Stachus
9	Düsseldorf	Altstadt
10	Stuttgart	Schlossplatz
11	Frankfurt	Mainufer
12	Berlin	Kudamm
13	Nürnberg	Zentrum
14	Bremen	Innenstadt
15	Hannover	Hauptbahnhof

```
15 rows in set (0,001 sec)
```

So sieht nun das Ergebnis aus. Nun muss geprüft werden, ob es für die Umstrukturierung in die dritte Normalform 3NF transitive Abhängigkeiten gibt.

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung;
```

Lieferungsnummer	Lieferzeitpunkt	Status_ID	Transporttyp_ID	Priorität_ID	Adresse_ID
1	2044-06-15	1	NULL	NULL	1
2	2044-06-16	2	NULL	NULL	2
3	2044-06-17	3	NULL	NULL	3
4	2044-06-18	4	NULL	NULL	4
5	2044-06-20	5	NULL	NULL	5
6	2044-06-21	3	NULL	NULL	5
7	2044-06-22	3	NULL	NULL	5
8	2044-06-23	3	NULL	NULL	5
9	2044-06-25	4	1	1	6
10	2044-06-27	1	2	2	NULL
11	NULL	3	3	3	7
12	2044-07-01	2	4	1	8
13	NULL	5	1	2	NULL
14	2044-07-05	NULL	5	1	9
15	2044-06-29	1	6	2	10
16	NULL	NULL	1	3	11
17	2044-07-03	4	2	1	NULL
18	NULL	5	7	2	12
19	NULL	NULL	8	3	NULL
20	2044-07-10	3	1	2	13
21	2044-07-12	NULL	NULL	NULL	14
22	2044-07-13	4	NULL	NULL	NULL
23	NULL	3	NULL	NULL	15
24	NULL	NULL	NULL	NULL	NULL

24 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferstatus;
```

Status_ID	Bezeichnung
1	unterwegs
2	geliefert
3	verspätet
4	geplant
5	in Bearbeitung

5 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Transporttyp;
```

Transporttyp_ID	Bezeichnung
1	Drohne
2	Elektro-LKW
3	Hybrid-LKW
4	Elektro-Van
5	Wasserstoff-LKW
6	Elektro-Scooter
7	Hybrid-Drohne
8	Autonomer LKW

8 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Versandkosten;
```

Transporttyp_ID	Kosten	Lieferungsnummer
1	19.99	9
1	12.99	13
1	14.50	16
1	10.99	20
2	29.99	10
2	17.89	17
3	24.50	11
4	15.75	12
5	22.30	14
6	9.99	15
7	11.99	18
8	31.50	19

12 rows in set (0,001 sec)

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferpriorität;
+-----+-----+
| Priorität_ID | Bezeichnung |
+-----+-----+
| 1 | 2 |
| 2 | 1 |
| 3 | 3 |
+-----+
3 rows in set (0,001 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferadresse;
+-----+-----+-----+
| Adresse_ID | Stadt | Straße |
+-----+-----+-----+
| 1 | Berlin | Alexanderplatz |
| 2 | Hamburg | Hafenstraße |
| 3 | München | Marienplatz |
| 4 | Frankfurt | Zeilstraße |
| 5 | Unbekannt | Unbekannt |
| 6 | Köln | Domplatz |
| 7 | Hamburg | Reeperbahn |
| 8 | München | Stachus |
| 9 | Düsseldorf | Altstadt |
| 10 | Stuttgart | Schlossplatz |
| 11 | Frankfurt | Mainufer |
| 12 | Berlin | Kudamm |
| 13 | Nürnberg | Zentrum |
| 14 | Bremen | Innenstadt |
| 15 | Hannover | Hauptbahnhof |
+-----+-----+-----+
15 rows in set (0,001 sec)
```

Potenzielle Probleme sind es, dass Status_ID von Transporttyp abhängen, dies jedoch bereits in Lieferstatus ausgelagert ist. Zudem ist Transporttyp_ID von Transporttyp abhängig, dies ist jedoch bereits in Transporttyp abhängig. Da wir bereits Transporttyp, Priorität und Adresse in separate Tabellen ausgelagert haben, sind wir nun in der dritten Normalform 3NF. Nach diesen Änderungen haben wir folgende saubere Tabellenstruktur: Lieferung (enthält Primärinformationen), Lieferadresse (enthält Stadt und Straße), Lieferstatus (enthält Status), Transporttyp (enthält Transportarten), Versandkosten (enthält unterschiedliche Kosten je Transporttyp), Lieferpriorität (enthält Prioritäten). Anhand dessen wird festgestellt, dass es keine Redundanzen mehr gibt, dass die Datenkonsistenz besser ist, und die Abfragen und der Speicherplatz optimiert ist.

Nun wollen wir die Tabelle Relation_Lieferung_Artikel bis zur dritten Normalform normalisieren. Diese Tabelle stellt eine n:m-Beziehung zwischen Lieferung und Artikel dar. Lieferungsnummer als Fremdschlüssel zur Lieferung, Artikelnummer als Fremdschlüssel zu Artikel, Liefermenge, Versandart und Expressversand.

```
MariaDB [hosakbari_db]> SELECT * FROM Relation_Lieferung_Artikel;
+-----+-----+-----+-----+-----+
| Lieferungsnummer | Artikelnummer | Liefermenge | Versandart | Expressversand |
+-----+-----+-----+-----+-----+
| 1 | 101 | NULL | NULL | NULL |
| 2 | 103 | NULL | NULL | NULL |
| 4 | 101 | NULL | NULL | NULL |
| 4 | 103 | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+
4 rows in set (0,001 sec)
```

Im ersten Schritt geht es um die Analyse der Normalformen. Was die erste Normalform 1NF betrifft, so ist zu betrachten, dass jede Spalte nur atomare Werte enthält, weshalb die 1NF schon bereit ist. Hinsichtlich der zweiten Normalform 2NF lässt sich sagen, dass die Nicht-Schlüsselattribute (Liefermenge, Versandart, Expressversand) nicht nur von der gesamten Primärschlüssel-Kombination abhängen. Beispiel hierzu ist es, dass Versandart und Expressversand nur von der Lieferungsnummer abhängen, nicht vom Artikel. Zur Lösung hierzu kann man eine Aufteilung in zwei Tabellen beanspruchen. Eine für die Beziehung Lieferung – Artikel als Lieferung_Artikel, sowie eine für die Versanddetails pro Lieferung Versanddetails. Hinsichtlich der dritten Normalform 3NF gibt es mögliche transitive Abhängigkeiten in Versanddetails. Als Beispiel hierzu hängt Expressversand oft von der Versandart ab. Zur Lösung hiervon kann man Versandart als separate Tabelle (Versandart) anlegen.

Als zweiten Schritt werden neue Tabellen für die 2NF erstellt.

1. Lösung 1: Trennung der n:m-Beziehung in eine neue Tabelle „Lieferung_Artikel“

```
CREATE TABLE Lieferung_Artikel (
    Lieferungsnummer INT,
    Artikelnummer INT,
    Liefermenge INT DEFAULT 1, -- Falls NULL, Standardwert setzen
    PRIMARY KEY (Lieferungsnummer, Artikelnummer),
    FOREIGN KEY (Lieferungsnummer) REFERENCES Lieferung(Lieferungsnummer),
    FOREIGN KEY (Artikelnummer) REFERENCES Artikel(Artikelnummer)
);
```

Diese Tabelle verwaltet nur die Beziehung zwischen Lieferung und Artikel. Liefermenge gehört zur spezifischen Lieferung eines Artikels, daher bleibt es hier.

2. Lösung 2: Erstellung einer neuen Tabelle Versanddetails

```
CREATE TABLE Versanddetails (
    Lieferungsnummer INT PRIMARY KEY,
    Versandart_ID INT,
    Expressversand BOOLEAN,
    FOREIGN KEY (Lieferungsnummer) REFERENCES Lieferung(Lieferungsnummer),
    FOREIGN KEY (Versandart_ID) REFERENCES Versandart(Versandart_ID)
);
```

```
MariaDB [hosakbari_db]> CREATE TABLE Versanddetails (
    Expressversand BO → Lieferungsnummer INT PRIMARY KEY,
    → Versandart_ID INT,
    → Expressversand BOOLEAN,
    → FOREIGN KEY (Lieferungsnummer) REFERENCES Lieferung(Lieferungsnummer),
    → FOREIGN KEY (Versandart_ID) REFERENCES Versandart(Versandart_ID)
    → );
ERROR 1005 (HY000): Can't create table 'hosakbari_db'.'Versanddetails' (errno: 150 "Foreign key constraint is incorrectly formed")
```

Wir bekommen hierbei die obige Fehlermeldung. Das liegt daran, weil wir zuerst die Tabelle Versandart erstellen müssen, und die Tabelle Versanddetails im Nachhinein erstellen müssen. So machen wir das, von vorne:

1. Tabelle Versandart zuerst erstellen

```
CREATE TABLE Versandart (
    Versandart_ID INT PRIMARY KEY AUTO_INCREMENT,
    Bezeichnung VARCHAR(50) NOT NULL
);
```

Nun existiert die Tabelle Versandart und kann in Fremdschlüsseln referenziert werden.

2. Danach erst Versanddetails erstellen

```
CREATE TABLE Versanddetails (
    Lieferungsnummer INT PRIMARY KEY,
    Versandart_ID INT,
    Expressversand BOOLEAN,
    FOREIGN KEY (Lieferungsnummer) REFERENCES Lieferung(Lieferungsnummer),
    FOREIGN KEY (Versandart_ID) REFERENCES Versandart(Versandart_ID)
);
```

Da die Tabelle Versandart bereits existiert, wird die Tabelle Versanddetails korrekt erstellt.

Im dritten Schritt erfolgt die Übertragung der bestehenden Daten

1. Übertragung der Lieferungs-Artikel-Beziehungen

```
INSERT INTO Lieferung_Artikel (Lieferungsnummer, Artikelnummer, Liefermenge)
SELECT Lieferungsnummer, Artikelnummer, COALESCE(Liefermenge, 1) FROM Relation_Lieferung_Artikel;
```

2. Übertragung der Versandinformationen in Versanddetails

```
INSERT INTO Versanddetails (Lieferungsnummer, Versandart_ID, Expressversand)
SELECT DISTINCT Lieferungsnummer, NULL, Expressversand FROM Relation_Lieferung_Artikel;
```

3. Übertragung der Versandarten

```
INSERT INTO Versandart (Bezeichnung)
```

```
SELECT DISTINCT Versandart FROM Relation_Lieferung_Artikel WHERE Versandart IS NOT NULL;
```

4. Aktualisierung der Versanddetails mit Versandart-IDs

```
UPDATE Versanddetails v
```

```
JOIN Relation_Lieferung_Artikel r ON v.Lieferungsnummer = r.Lieferungsnummer
```

```
JOIN Versandart va ON r.Versandart = va.Bezeichnung
```

```
SET v.Versandart_ID = va.Versandart_ID;
```

Im vierten Schritt entfernen wir die alte Tabelle

```
DROP TABLE Relation_Lieferung_Artikel;
```

```
MariaDB [hosakbari_db]> SELECT * FROM Lieferung_Artikel;
```

Lieferungsnummer	Artikelnummer	Liefermenge
1	101	1
2	103	1
4	101	1
4	103	1

```
4 rows in set (0,001 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Versanddetails;
```

Lieferungsnummer	Versandart_ID	Expressversand
1	NULL	NULL
2	NULL	NULL
4	NULL	NULL

```
3 rows in set (0,001 sec)
```

```
MariaDB [hosakbari_db]> SELECT * FROM Versandart;
```

```
Empty set (0,001 sec)
```

So wie oben sieht nun das endgültige Ergebnis aus. Wie man erkennen kann gibt es erstens für die 1NF keine mehrfachen Werten in Spalten, zum zweiten hängen alle Nicht-Schlüsselattribute nur von der gesamten Primärschlüssel-Kombination ab. Zum dritten für die 3NF gibt es keine transitiven Abhängigkeiten mehr.

Aufgabe 16.01.2025: - Erzeugen Sie 5 virtuelle Views (Datensichten); - Schreiben Sie die formulierten Unterabfragen mit Hilfe von SQL; - Anschließend einen Screenshot hinzufügen; - Problematik „Logistik in 20 Jahren“

Bei dieser Aufgabenstellung sollen virtuelle Views (Sichten) erstellt werden. Hierzu werden die Unterabfragen in SQL-Views umgewandelt und in einer strukturierten Art und Weise präsentiert.

- Virtueller View 1: Übersicht aller Lieferungen mit Adressen und Status

```
CREATE VIEW Lieferung_Übersicht AS
```

```
SELECT
```

```
I.Lieferungsnummer,  
I.Lieferzeitpunkt,  
ls.Bezeichnung AS Lieferstatus,  
lt.Bezeichnung AS Transportmittel,  
lp.Bezeichnung AS Priorität,  
la.Stadt,  
la.Straße
```

```
FROM Lieferung I
```

```
LEFT JOIN Lieferstatus ls ON I.Status_ID = ls.Status_ID  
LEFT JOIN Transporttyp lt ON I.Transporttyp_ID = lt.Transporttyp_ID  
LEFT JOIN Lieferpriorität lp ON I.Priorität_ID = lp.Priorität_ID  
LEFT JOIN Lieferadresse la ON I.Adresse_ID = la.Adresse_ID;
```

Es wird eine Ansicht erstellt, die eine umfassende Übersicht über alle Lieferungen bietet. Die angezeigten Informationen umfassen die Lieferungsnummer, den Lieferzeitpunkt, den Status der Lieferung, das verwendete Transportmittel, die Priorität der Lieferung sowie die Stadt und die Straße der jeweiligen Lieferadresse. Dazu werden die Tabellen „Lieferung“, „Lieferstatus“, „Transporttyp“, „Lieferpriorität“ und „Lieferadresse“ über ihre Fremdschlüsselelemente miteinander verknüpft. Es werden alle Einträge aus der „Lieferung“-Tabelle angezeigt, auch wenn zugehörige Einträge in den anderen Tabellen fehlen, da LEFT JOINS verwendet werden.

Hier sieht man einmal das Ergebnis

Lieferungsnummer	Lieferzeitpunkt	Lieferstatus	Transportmittel	Priorität	Stadt	Straße
1	2044-06-15	unterwegs	NULL	NULL	Berlin	Alexanderplatz
2	2044-06-16	geliefert	NULL	NULL	Hamburg	Hafenstraße
3	2044-06-17	verspätet	NULL	NULL	München	Marienplatz
4	2044-06-18	geplant	NULL	NULL	Frankfurt	Zeilstraße
5	2044-06-20	in Bearbeitung	NULL	NULL	Unbekannt	Unbekannt
6	2044-06-21	verspätet	NULL	NULL	Unbekannt	Unbekannt
7	2044-06-22	verspätet	NULL	NULL	Unbekannt	Unbekannt
8	2044-06-23	verspätet	NULL	NULL	Unbekannt	Unbekannt
9	2044-06-25	geplant	Drohne	2	Köln	Domplatz
10	2044-06-27	unterwegs	Elektro-LKW	1	NULL	NULL
11	NULL	verspätet	Hybrid-LKW	3	Hamburg	Reeperbahn
12	2044-07-01	geliefert	Elektro-Van	2	München	Stachus
13	NULL	in Bearbeitung	Drohne	1	NULL	NULL
14	2044-07-05	NULL	Wasserstoff-LKW	2	Düsseldorf	Altstadt
15	2044-06-29	unterwegs	Elektro-Scooter	1	Stuttgart	Schlossplatz
16	NULL	NULL	Drohne	3	Frankfurt	Mainufer
17	2044-07-03	geplant	Elektro-LKW	2	NULL	NULL
18	NULL	in Bearbeitung	Hybrid-Drohne	1	Berlin	Kudamm
19	NULL	NULL	Autonomer LKW	3	NULL	NULL
20	2044-07-10	verspätet	Drohne	1	Nürnberg	Zentrum
21	2044-07-12	NULL	NULL	NULL	Bremen	Innenstadt
22	2044-07-13	geplant	NULL	NULL	NULL	NULL
23	NULL	verspätet	NULL	NULL	Hannover	Hauptbahnhof
24	NULL	NULL	NULL	NULL	NULL	NULL

24 rows in set (0,002 sec)

- Virtueller View 2: Nachhaltige Lagerhäuser mit erneuerbaren Energiequellen

CREATE VIEW Nachhaltige_Lagerhäuser AS

SELECT

```

    lh.Lager_ID,
    lh.Stadt,
    lh.Straße,
    e.Bezeichnung AS Energiequelle
FROM Lagerhaus lh
JOIN Energiequelle e ON lh.Energiequelle_ID = e.Energiequelle_ID
WHERE e.Bezeichnung IN ('Solar', 'Wind', 'Erneuerbar');
```

Es wird eine Ansicht erstellt, die alle Lagerhäuser anzeigt, die erneuerbare Energiequellen wie Solar, Wind oder Erneuerbare nutzen. Angezeigt werden die Lagerhaus-ID, die Stadt und Straße des Standorts sowie die Bezeichnung der Energiequelle. Die Verbindung zwischen den Tabellen „Lagerhaus“ und „Energiequelle“ erfolgt über die Fremdschlüssel „Energiequelle_ID“. Nur Lagerhäuser, die eine dieser drei Energiequellen verwenden, werden gefiltert und in der Ansicht dargestellt.

Lager_ID	Stadt	Straße	Energiequelle
1	Berlin	Lagerstraße 12	Solar
2	Hamburg	Hafenlager 23	Wind
3	München	Südallee 45	Solar
4	Frankfurt	Westhafen 8	Solar
5	Leipzig	Zentrallager	Erneuerbar
6	Stuttgart	Hauptlager 1	Erneuerbar
7	Düsseldorf	Logistikzentrum	Erneuerbar
8	Dresden	Verteilzentrum	Erneuerbar

8 rows in set (0,002 sec)

- Virtueller View 3: Lieferungen mit höchsten Versandkosten

```

CREATE VIEW Teuerste_Lieferungen AS
SELECT
    v.Lieferungsnummer,
    l.Lieferzeitpunkt,
    lt.Bezeichnung AS Transportmittel,
    v.Kosten AS Versandkosten
FROM Versandkosten v
JOIN Lieferung l ON v.Lieferungsnummer = l.Lieferungsnummer
JOIN Transporttyp lt ON l.Transporttyp_ID = lt.Transporttyp_ID
ORDER BY v.Kosten DESC
LIMIT 10;

```

Es wird eine Ansicht erstellt, die die zehn teuersten Lieferungen anhand der Versandkosten anzeigt. Die Liste enthält die Lieferungsnummer, den Lieferzeitpunkt, das verwendete Transportmittel sowie die jeweiligen Versandkosten. Dazu werden die Tabellen „Versandkosten“, „Lieferung“ und „Transporttyp“ miteinander verknüpft. Die Einträge werden nach den Versandkosten in absteigender Reihenfolge sortiert, und nur die obersten zehn Ergebnisse werden angezeigt.

```
MariaDB [hosakbari_db]> SELECT * FROM Teuerste_Lieferungen;
```

Lieferungsnummer	Lieferzeitpunkt	Transportmittel	Versandkosten
19	NULL	Autonomer LKW	31.50
10	2044-06-27	Elektro-LKW	29.99
11	NULL	Hybrid-LKW	24.50
14	2044-07-05	Wasserstoff-LKW	22.30
9	2044-06-25	Drohne	19.99
17	2044-07-03	Elektro-LKW	17.89
12	2044-07-01	Elektro-Van	15.75
16	NULL	Drohne	14.50
13	NULL	Drohne	12.99
18	NULL	Hybrid-Drohne	11.99

```
10 rows in set (0,002 sec)
```

- Virtueller View 4: Durchschnittliche Versandkosten pro Transporttyp

```

CREATE VIEW Durchschnittliche_Versandkosten AS
SELECT
    t.Bezeichnung AS Transportmittel,
    ROUND(AVG(v.Kosten), 2) AS Durchschnittskosten
FROM Versandkosten v
JOIN Transporttyp t ON v.Transporttyp_ID = t.Transporttyp_ID
GROUP BY t.Bezeichnung;

```

Es wird eine Ansicht erstellt, die die durchschnittlichen Versandkosten für jedes Transportmittel berechnet. Angezeigt werden das Transportmittel und die berechneten Durchschnittskosten, wobei diese auf zwei Nachkommastellen gerundet sind. Die Tabellen „Versandkosten“ und „Transporttyp“ werden über den Fremdschlüssel „Transporttyp_ID“ verknüpft, und die Durchschnittswerte werden mit der Aggregatfunktion AVG berechnet und anschließend mit ROUND gerundet.

```
MariaDB [hosakbari_db]> SELECT * FROM Durchschnittliche_Versandkosten;
+-----+-----+
| Transportmittel | Durchschnittskosten |
+-----+-----+
| Autonomer LKW | 31.50 |
| Drohne | 14.62 |
| Elektro-LKW | 23.94 |
| Elektro-Scooter | 9.99 |
| Elektro-Van | 15.75 |
| Hybrid-Drohne | 11.99 |
| Hybrid-LKW | 24.50 |
| Wasserstoff-LKW | 22.30 |
+-----+-----+
8 rows in set (0,002 sec)
```

- Virtueller View 5: Wartungsstatus der Fahrzeuge

```
CREATE VIEW Fahrzeuge_Wartung AS
SELECT
    f.Fahrzeug_ID,
    ft.Bezeichnung AS Fahrzeugtyp,
    w.Wartungsstatus
FROM Fahrzeug f
JOIN Fahrzeugtyp ft ON f.Fahrzeugtyp_ID = ft.Fahrzeugtyp_ID
LEFT JOIN Wartung w ON f.Fahrzeug_ID = w.Fahrzeug_ID;
```

Es wird eine Ansicht erstellt, die den aktuellen Wartungsstatus aller Fahrzeuge zusammen mit ihrer Fahrzeug-ID und dem jeweiligen Fahrzeugtyp anzeigt. Dazu werden die Tabellen „Fahrzeug“, „Fahrzeugtyp“ und „Wartung“ verknüpft. Auch Fahrzeuge ohne Wartungsstatus werden angezeigt, da ein LEFT JOIN zwischen „Fahrzeug“ und „Wartung“ verwendet wird, sodass alle Fahrzeuge in der Ansicht enthalten sind.

```
MariaDB [hosakbari_db]> SELECT * FROM Fahrzeuge_Wartung;
+-----+-----+-----+
| Fahrzeug_ID | Fahrzeugtyp | Wartungsstatus |
+-----+-----+-----+
| 1 | Elektro-LKW | NULL |
| 3 | Elektro-Van | NULL |
| 5 | Autonomer LKW | NULL |
| 6 | Wasserstoff-LKW | NULL |
| 9 | Elektro-LKW | NULL |
| 11 | Hybrid-LKW | NULL |
| 12 | Drohne | NULL |
| 13 | Elektro-Van | NULL |
| 15 | Autonomer LKW | NULL |
| 16 | Wasserstoff-LKW | NULL |
| 18 | Hybrid-Drohne | NULL |
+-----+-----+-----+
11 rows in set (0,035 sec)
```

- Virtueller View 6: Durchschnittliche Liefermenge pro Artikel und Transportmittel

```
CREATE VIEW Durchschnittliche_Liefermenge AS
SELECT
    a.Bezeichnung AS Artikelname,
    t.Bezeichnung AS Transportmittel,
    ROUND(AVG(la.Liefermenge), 2) AS Durchschnittsmenge
FROM Lieferung_Artikel la
JOIN Artikel a ON la.Artikelnummer = a.Artikelnummer
JOIN Lieferung l ON la.Lieferungsnummer = l.Lieferungsnummer
```

```
LEFT JOIN Transporttyp t ON l.Transporttyp_ID = t.Transporttyp_ID
GROUP BY a.Bezeichnung, t.Bezeichnung;
```

Es wird eine Ansicht erstellt, die die durchschnittliche Liefermenge für jeden Artikel und jedes Transportmittel berechnet. Die Ansicht zeigt den Artikelnamen, das verwendete Transportmittel und die durchschnittliche Liefermenge, die auf zwei Nachkommastellen gerundet ist. Die Tabellen „Lieferung_Artikel“, „Artikel“, „Lieferung“ und „Transporttyp“ werden miteinander verknüpft, und die Aggregatfunktion AVG wird für die Berechnung der durchschnittlichen Liefermenge verwendet.

```
MariaDB [hosakbari_db]> SELECT * FROM Durchschnittliche_Liefermenge;
+-----+-----+-----+
| Artikelname | Transportmittel | Durchschnittsmenge |
+-----+-----+-----+
| Smartphone-Modell X | NULL | 1.00 |
| Tablet-Modell Z | NULL | 1.00 |
+-----+-----+-----+
2 rows in set (0,031 sec)
```

- Virtueller View 7: Lagerhäuser mit der höchsten Gesamtkapazität

```
CREATE VIEW Größte_Lagerhäuser AS
SELECT
    lh.Stadt,
    lh.Straße,
    lk.Kapazität,
    RANK() OVER (ORDER BY lk.Kapazität DESC) AS Rang
FROM Lagerhaus lh
JOIN Lagerkapazität lk ON lh.Lagerkapazität_ID = lk.Lagerkapazität_ID;
```

Es wird eine Ansicht erstellt, die die größten Lagerhäuser anhand ihrer Kapazität zeigt. Angezeigt werden die Stadt und Straße der Lagerhäuser, ihre Kapazität sowie ein Rang, der auf der Kapazität basiert. Die Tabellen „Lagerhaus“ und „Lagerkapazität“ werden über den Fremdschlüssel „Lagerkapazität_ID“ verbunden, und der Rang wird mit der analytischen Funktion RANK berechnet, die die Lagerhäuser absteigend nach ihrer Kapazität sortiert.

```
MariaDB [hosakbari_db]> SELECT * FROM Größte_Lagerhäuser;
+-----+-----+-----+-----+
| Stadt | Straße | Kapazität | Rang |
+-----+-----+-----+-----+
| Frankfurt | Westhafen 8 | 100000 | 1 |
| Hamburg | Hafenlager 23 | 75000 | 2 |
| Dresden | Verteilzentrum | 60000 | 3 |
| Stuttgart | Hauptlager 1 | 55000 | 4 |
| Berlin | Lagerstraße 12 | 50000 | 5 |
| Düsseldorf | Logistikzentrum | 50000 | 5 |
| Leipzig | Zentrallager | 45000 | 7 |
| München | Südallee 45 | 35000 | 8 |
+-----+-----+-----+-----+
8 rows in set (0,050 sec)
```

- Virtueller View 8: Lieferstatus-Statistik für das letzte Jahr

```
CREATE VIEW Lieferstatus_Statistik AS
SELECT
    ls.Bezeichnung AS Lieferstatus,
    COUNT(l.Lieferungsnummer) AS Anzahl_Lieferungen,
    ROUND(COUNT(l.Lieferungsnummer) * 100.0 /
        (SELECT COUNT(*) FROM Lieferung WHERE Lieferzeitpunkt >= '2043-01-01'), 2) AS Prozent
```

```

FROM Lieferung l
JOIN Lieferstatus ls ON l.Status_ID = ls.Status_ID
WHERE l.Lieferzeitpunkt >= '2043-01-01'
GROUP BY ls.Bezeichnung;

```

Es wird eine Ansicht erstellt, die eine Statistik über die Lieferstatus anzeigt. Für jeden Lieferstatus werden die Anzahl der Lieferungen und deren prozentualer Anteil an allen Lieferungen seit dem Jahr 2043 berechnet. Die Tabellen „Lieferung“ und „Lieferstatus“ werden miteinander verknüpft, und die Aggregatfunktion COUNT zählt die Lieferungen. Die prozentuale Berechnung erfolgt durch Division der Anzahl der Lieferungen je Status durch die Gesamtzahl der Lieferungen seit 2043.

```

MariaDB [hosakbari_db]> SELECT * FROM Lieferstatus_Statistik;
+-----+-----+-----+
| Lieferstatus | Anzahl_Lieferungen | Prozent |
+-----+-----+-----+
| geliefert    |              2 |   11.76 |
| geplant      |              4 |   23.53 |
| in Bearbeitung |          1 |   5.88  |
| unterwegs    |              3 |   17.65 |
| verspätet    |              5 |   29.41 |
+-----+-----+-----+
5 rows in set (0,025 sec)

```

- Virtueller View 9: Durchschnittliche Lieferzeit pro Transportmittel

```

CREATE VIEW Durchschnittliche_Lieferzeit AS
SELECT
    t.Bezeichnung AS Transportmittel,
    ROUND(AVG(DATEDIFF(l.Lieferzeitpunkt, NOW())), 2) AS Durchschnittliche_Tage
FROM Lieferung l
JOIN Transporttyp t ON l.Transporttyp_ID = t.Transporttyp_ID
WHERE l.Lieferzeitpunkt IS NOT NULL
GROUP BY t.Bezeichnung;

```

Es wird eine Ansicht erstellt, die die durchschnittliche Lieferzeit in Tagen für jedes Transportmittel berechnet. Angezeigt werden das Transportmittel und die berechnete durchschnittliche Lieferzeit, die auf zwei Nachkommastellen gerundet ist. Die Tabellen „Lieferung“ und „Transporttyp“ werden über den Fremdschlüssel „Transporttyp_ID“ verknüpft, und die Differenz zwischen dem Lieferzeitpunkt und dem aktuellen Datum wird mit der Funktion DATEDIFF berechnet und anschließend mit AVG gemittelt.

```

MariaDB [hosakbari_db]> SELECT * FROM Durchschnittliche_Lieferzeit;
+-----+-----+
| Transportmittel | Durchschnittliche_Tage |
+-----+-----+
| Drohne          |       7080.50 |
| Elektro-LKW     |       7078.00 |
| Elektro-Scooter |       7077.00 |
| Elektro-Van     |       7079.00 |
| Wasserstoff-LKW |       7083.00 |
+-----+-----+
5 rows in set (0,007 sec)

```

- Virtueller View 10: Fahrzeuge, die keiner Lieferung zugeordnet sind

```

CREATE VIEW Unbenutzte_Fahrzeuge AS
SELECT
    f.Fahrzeug_ID,
    ft.Bezeichnung AS Fahrzeugtyp

```

```
FROM Fahrzeug f
LEFT JOIN Fahrzeugtyp ft ON f.Fahrzeugtyp_ID = ft.Fahrzeugtyp_ID
WHERE f.Lieferung_Lieferungsnummer IS NULL;
```

Es wird eine Ansicht erstellt, die alle Fahrzeuge anzeigt, die aktuell keiner Lieferung zugeordnet sind. Angezeigt werden die Fahrzeug-ID und der Fahrzeugtyp. Die Tabellen „Fahrzeug“ und „Fahrzeugtyp“ werden miteinander verknüpft, und nur Fahrzeuge, bei denen der Fremdschlüssel „Lieferung_Lieferungsnummer“ NULL ist, werden in der Ansicht angezeigt.

```
MariaDB [hosakbari_db]> SELECT * FROM Unbenutzte_Fahrzeuge;
+-----+-----+
| Fahrzeug_ID | Fahrzeugtyp |
+-----+-----+
|      9 | Elektro-LKW
|     13 | Elektro-Van
|     14 | NULL
|     17 | NULL
|     18 | Hybrid-Drohne
+-----+-----+
5 rows in set (0,021 sec)
```

Hier sieht man übrigens nochmal alle Tabellen

```
MariaDB [hosakbari_db]> SHOW TABLES;
+-----+
| Tables_in_hosakbari_db |
+-----+
| Artikel
| Beste_CO2_Fahrzeuge
| Durchschnittliche_Liefermenge
| Durchschnittliche_Lieferzeit
| Durchschnittliche_Versandkosten
| Energiequelle
| Express_High_Priority
| Expresslieferungen
| Fahrzeug
| Fahrzeuge_Wartung
| Fahrzeugtyp
| Fahrzeugtyp_Ladezeit
| Größte_Lagerhäuser
| Lagerbestand
| Lagerhaus
| Lagerhaus_Eigenschaften
| Lagerkapazität
| Lieferadresse
| Lieferpriorität
| Lieferstatus
| Lieferstatus_Statistik
| Lieferung
| Lieferung_Artikel
| Lieferung_Übersicht
| Nachhaltige_Lagerhäuser
| Produkttyp
| Sicherheitsstufe
| Teuerste_Lieferungen
| Transporttyp
| Unbenutzte_Fahrzeuge
| Versandart
| Versanddetails
| Versandkosten
| Wartung
+-----+
34 rows in set (0,001 sec)
```

Quellenverzeichnis

- <https://www.purestorage.com/de/knowledge/what-is-mariadb.html>
- <https://files.oaiusercontent.com/file-FstcwkW3ujg3psvxNVzmwrcd?se=2025-02-13T16%3A42%3A22Z&sp=r&sv=2024-08-04&sr=b&rscc=max-age%3D604800%2C%20immutable%2C%20private&rscd=attachment%3B%20filename%3D02ed56b2-fb96-4e6d-a8fc-4c1f5d2a2ca4.webp&sig=/6p8ZNeLinr%2Bu13x48b/EjwntRUbjYIYKF1RsWsDSFg%3D>
- <https://files.oaiusercontent.com/file-5TOLrn3OXGjllOHH8WI79xE4?se=2025-02-13T16%3A42%3A22Z&sp=r&sv=2024-08-04&sr=b&rscc=max-age%3D604800%2C%20immutable%2C%20private&rscd=attachment%3B%20filename%3Df4003b06-40b3-4e45-9754-0df5a4d183c0.webp&sig=XUSqKTXHbih1pfDjHtjv5wicmA/CBFMPZF7L5vRTIBg%3D>
- <https://files.oaiusercontent.com/file-DTGJoDyIHanN4p3hya0aXN8p?se=2025-02-13T16%3A42%3A22Z&sp=r&sv=2024-08-04&sr=b&rscc=max-age%3D604800%2C%20immutable%2C%20private&rscd=attachment%3B%20filename%3Dd3c8fe2d-98fa-41c8-a9e8-04c26ace4fd5.webp&sig=BWfM/JFsjNxN4fM6%2BJO0SshAke2qsE4d%2BZMg4b4BhvY%3D>

DATENBANKEN 1

SQL

Prof. Dr. Nadija Petram

B.Sc. Wirtschaftsinformatik – Hossein Akbari – Datenbanken 1 – SQL
Wintersemester 2024/25 - Hochschule Bremerhaven

GLIEDERUNG

- SQL - was ist das?
- Definition und Ursprünge
- Theoretische Hinführung: Das relationale Datenbankmodell
- MySQL – eines der beliebtesten Datenbanksysteme
- Die ersten Schritte: Datenbank anlegen und Tabellen erstellen
- Werte einfügen, bearbeiten und abfragen
- Beziehungen zwischen Tabellen mit Fremdschlüssel?
- Praktische Anwendung von SQL
- Zusammenfassung und Schluss

EINFÜHRUNG

- SQL - was ist das?
 - Definition
 - SQL - "Structured Query Language" - Datenbanksprache
 - Kommunikationsmittel zwischen Anwender und Datenbank
 - Keine Programmiersprache
 - Funktionen von SQL
 - Datenbankstruktur erstellen
 - Tabellen anlegen
 - Werte eintragen
 - Abfrageergebnisse sortieren + strukturieren
- Funktionen von SQL (Fort.)
 - Manuell und über Computerprogramm

EINFÜHRUNG

- Definition und Ursprünge
 - Definition einer Datenbank
 - System für Datenverwaltung
 - Aufgabe: Informationen ...
 - Dauerhaft, sicher, effizient, abspeichern
 - Datenbank aus zwei Ebenen:
 - Physische Ebene des Datenspeichers
 - Festplatte oder SSD (Hardware)
 - Datenbankmanagementsystems (DBMS) - (Software)
- Rolle des DBMS
 - Verwaltung des physischen Datenspeichers
 - Schnittstelle zwischen Daten und Anwender
 - SQL als Datenbanksprache
- DBMS ermöglicht:
 - Eingabe von Befehlen
 - Direkt oder durch Computerprogramm

EINFÜHRUNG

- Ursprünge
 - Entstehung - 70 Jahren
 - Chamberlin und Boyce - entwickeln SQL bei IBM
 - Basierend auf rel. Datenbankmodell von Edgar F. Codd.
 - Vorbereitung und Standardisierung
 - Weltweit populär in den 80ern
 - 1986 - Erster SQL-Standard verabschiedet
 - Seit dem - Meist benutzte DB-Sprache

DAS RELATIONALE DATENBANKMODELL

- Was ist ein Datenbankmodell?
 - Struktur für Datenablage einer Datenbank
 - Vorgegeben durch DBMS
 - Beeinflusst Datenbankzugriff - erfordert Datenbanksprache
 - Bsp.: SQL für Verwaltung der Daten

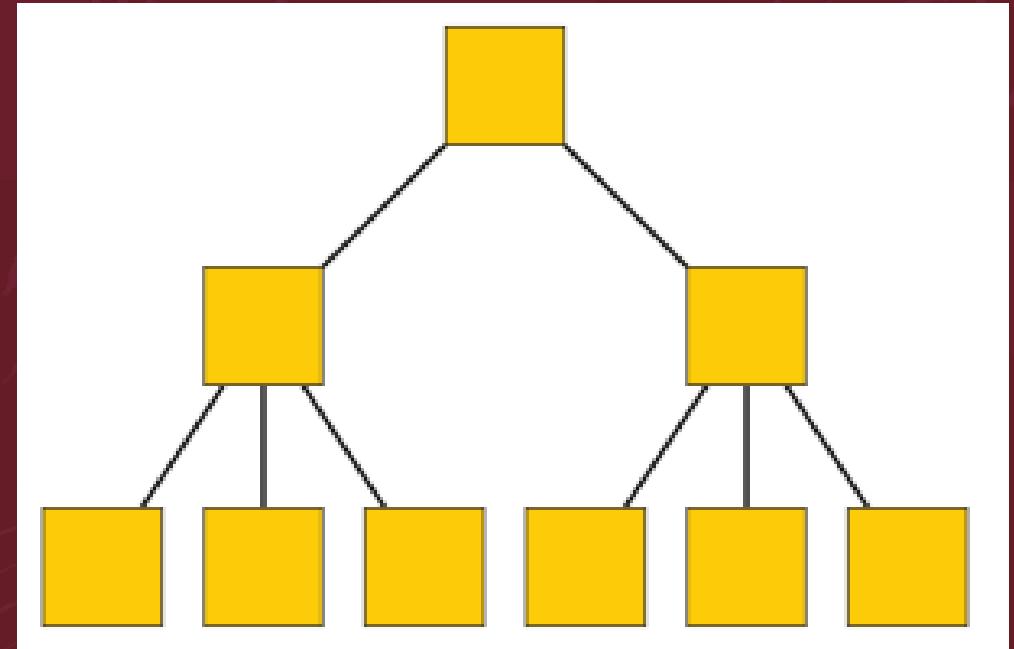
DAS RELATIONALE DATENBANKMODELL

GRUNDLEGE EIGENSCHAFTEN DER DATENBANKMODELLE:

Vorstellung alternativer Datenbankmodelle:

Hierarchisches Datenbankmodell:

- Baumstruktur mit Ausgangselement (Wurzel) und Kind-Elementen
- Einschränkung: Nur Beziehungen zwischen Eltern- und Kind-Elementen, keine über mehrere Elementen



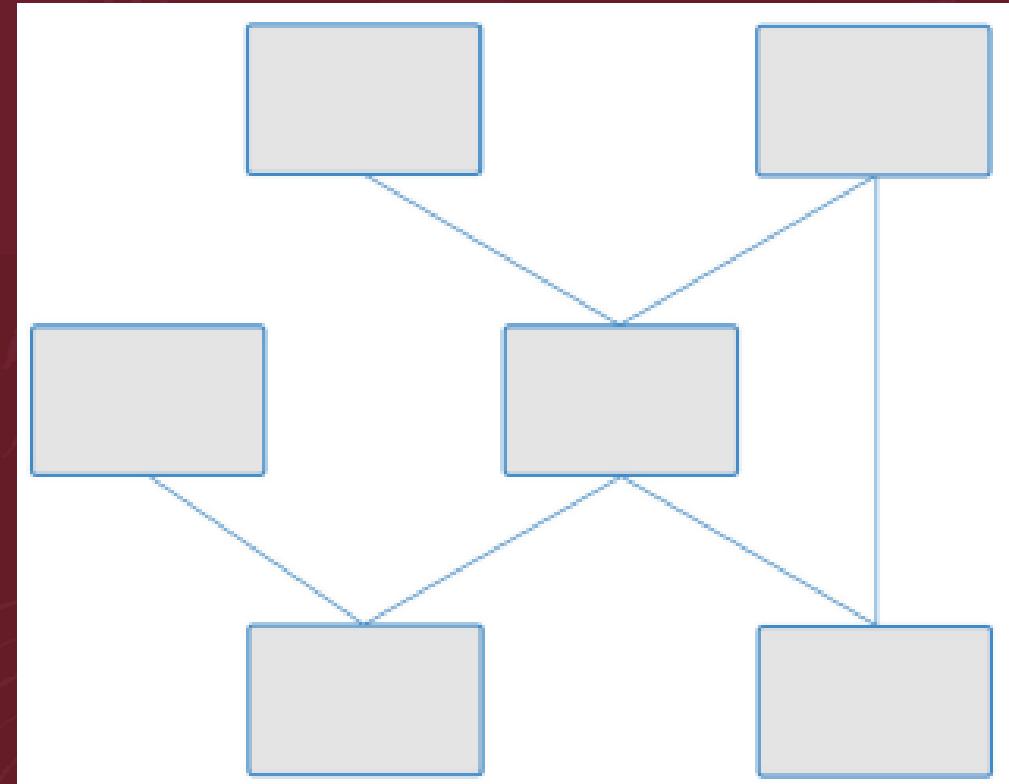
DAS RELATIONALE DATENBANKMODELL

GRUNDLEGE EIGENSCHAFTEN DER DATENBANKMODELLE:

Vorstellung alternativer Datenbankmodelle:

Netzwerkdatenbankmodell:

- Entwickelt gleichzeitig mit dem relationalen Modell
- Basiert auf dem hierarchischen Modell
 - Ziel: Dessen Einschränkungen ausgleichen
 - Komplexe Beziehungen zwischen Einträgen
 - Keine klaren Suchwege



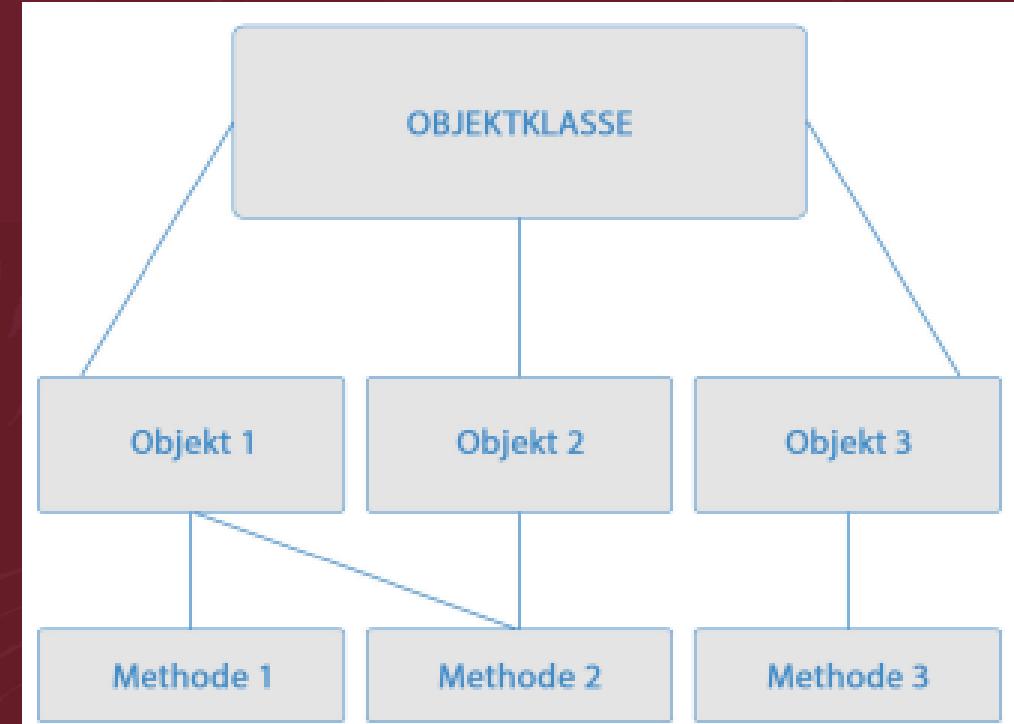
DAS RELATIONALE DATENBANKMODELL

GRUNDLAGE EIGENSCHAFTEN DER DATENBANKMODELLE:

Vorstellung alternativer Datenbankmodelle:

Objektorientierter Datenbank

- Definiert Objekte mit zugeordneten Attributen
- Übertragung von OOP auf Datenbank möglich
- Objektdatenbank ermöglicht Definition von Methoden



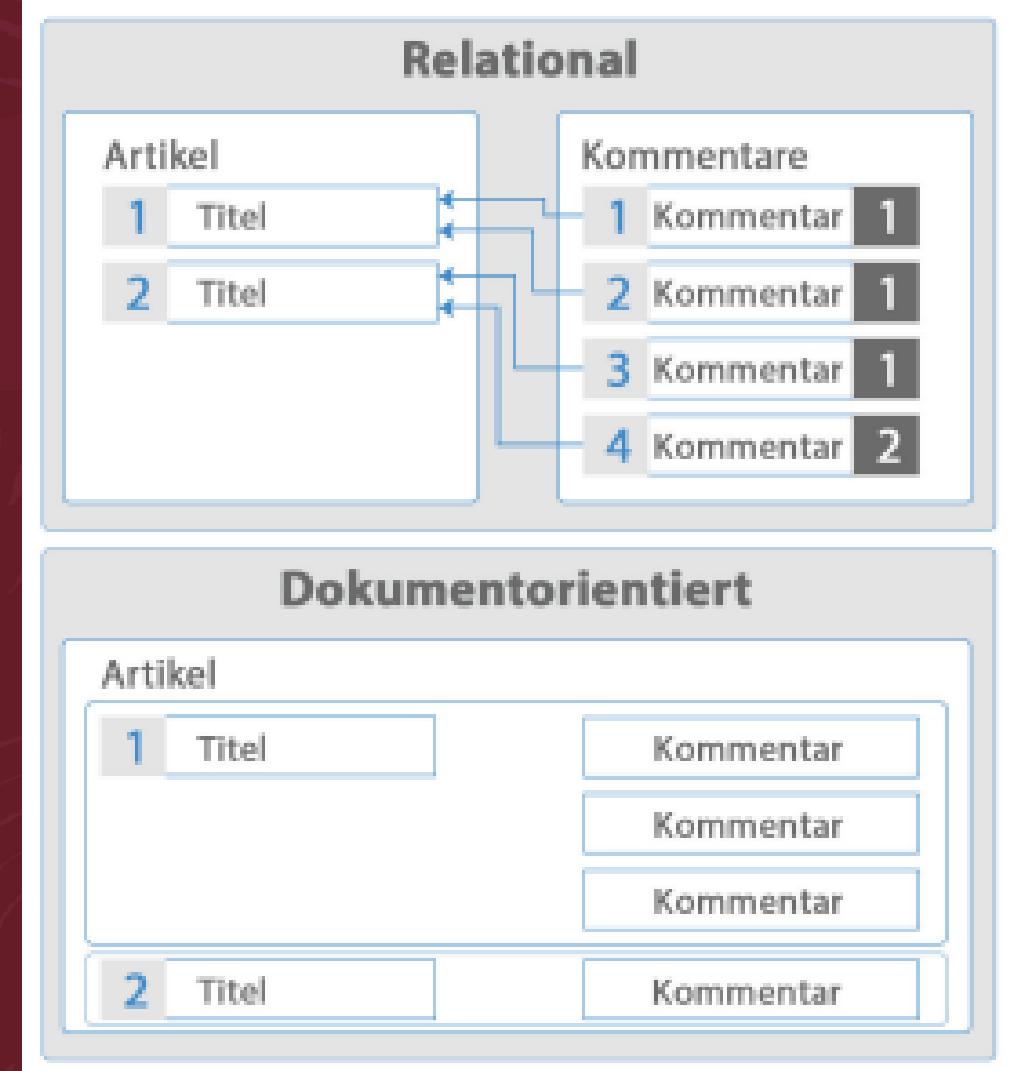
DAS RELATIONALE DATENBANKMODELL

GRUNDLEGE EIGENSCHAFTEN DER DATENBANKMODELLE:

Vorstellung alternativer Datenbankmodelle:

Objektrelationaler Datenbank

- Mischform zwischen objektorientierter und relationaler Datenbank
- Erlaubt Erzeugung von Objekten und Definition von Relationen

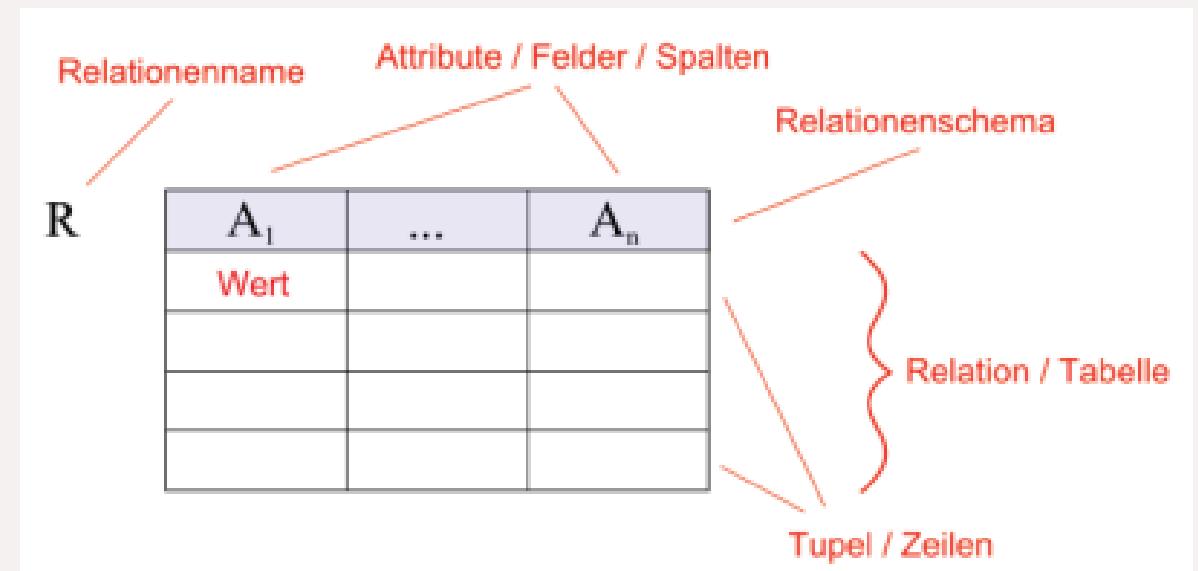


DAS RELATIONALE DATENBANKMODELL

GRUNDLEGENDE EIGENSCHAFTEN DER DATENBANKMODELLE:

Relationale Datenbankmodelle im Detail

- Basieren auf Tabellen
- Tupel/Zeilen und Attribute:
 - Zeilen: Daten zum Gegenstand/Person
 - Felder/Spalten einer Zeile: Enthalten Attributwerte
 - Spalten: Definieren Struktur einer Tabelle
 - Schlüssel: Jeder Datensatz muss eindeutig identifizierbar sein - Durch Verwendung von Schlüssel
 - Beziehungen zwischen Tabellen: Sind möglich und wichtig
 - Beispiel: Mitgliederliste und Angebote - Beziehung über Schlüssel



MySQL - eines der beliebtesten Datenbanksysteme

- 1994: Gestartet von Michael Widenius, David Axmark und Allan Larson
- Ziel: Quelloffene Alternative für Datenbankverwaltung mit SQL
- Erfolg im Internet:
 - Mit Zunahme dynamischer Webseiten – Steigt Bedarf an geeigneter Datenbanksoftware
 - MySQL wurde bekannt als kostengünstige Open-Source-Alternative
- Entstehung: Begriff LAMP (Linux, Apache, MySQL, PHP)
 - Möglichkeit, einfach und effizient dynamische Webseiten bauen
 - Hierdurch MySQL bekannt – heute
- Nutzung von Millionen Internetangebote: Darunter Facebook, YouTube, Twitter, Flickr
- 2008: Übernommen von Sun Microsystems
- 2009: Ab dann zu MariaDB weiterentwickelt
- Andere beliebte Datenbanksysteme: PostgreSQL, Microsoft SQL-Server, Oracle Database, ...

ERSTE SCHRITTE: Datenbank anlegen und Tabellen erstellen

- Datenbank anlegen
 1. Überlegung der Datenstruktur: Welche Daten soll in Datenbank
 2. Anzahl der Datenbanken festlegen:
 - Eine Datenbank für alle Daten? oder
 - Mehrere Datenbanken für verschiedene Teilbereiche?
 3. SQL-Befehl: CREATE DATABASE testDB2;
 4. Überprüfen: SHOW DATABASE;
 5. Danach in Datenbank Tabelle erstellen ...

ERSTE SCHRITTE: Datenbank anlegen und Tabellen erstellen

- Tabelle gestalten
 1. Tabelle innerhalb einer Datenbank anlegen
 1. Befehl, um Datenbank auswählen: USE vereinDB;
 2. Tabelle erstellen und Spalten definieren:
- Hinweise:
 - Struktur der Tabelle wird durch Spalten bestimmt
 - Datentyp der Spalte soll passen zu den gespeicherten Daten sein
 - Eindeutige Bezeichner für Spalten verwenden
 - CREATE TABLE IF NOT EXISTS, stellt sicher, Tabellen-Dopplungen zu vermeiden

```
1 CREATE TABLE IF NOT EXISTS mitglieder(  
2     mitglieder_id INT,  
3     mitglieder_vorname VARCHAR(20),  
4     mitglieder_name VARCHAR(20),  
5     mitglieder_telefon INT,  
6     mitglieder_alter INT  
7 );
```

ERSTE SCHRITTE: Datenbank anlegen und Tabellen erstellen

- Primärschlüssel und weitere Bedingungen und Attribute
 - Primärschlüssel (PRIMARY KEY)
 - Erzeugt Primärschlüssel für Spalte
 - Jeder Wert der Spalte ist eindeutig - kommt nicht wiederholt vor
 - NOT NULL Bedingung
 - Stellt sicher, dass der Wert der Spalte ungleich 0 ist
 - UNIQUE Bedingung
 - Stellt sicher, dass der Wert der Spalte eindeutig ist
 - AUTO_INCREMENT Attribut
 - Inkrementiert den Wert (normalerweise 1)
 - Wird oft für Primärschlüssel benutzt
 - UNSIGNED Attribut:
 - Stellt sicher, dass das Feld nur nicht-negative Zahlen enthält

```
10 CREATE TABLE IF NOT EXISTS mitglieder2(
11     mitglieder2_id INT PRIMARY KEY AUTO_INCREMENT,
12     mitglieder2_vorname VARCHAR(20) NOT NULL,
13     mitglieder2_name VARCHAR(20) NOT NULL,
14     mitglieder2_telefon INT,
15     mitglieder2_alter TINYINT UNSIGNED,
16     mitglieder2_nutzername VARCHAR(20) UNIQUE
17 );
```

ERSTE SCHRITTE: Datenbank anlegen und Tabellen erstellen

Nachträgliche Änderungen der Tabelle

- Spalten hinzufügen (ADD)
 - Befehl: ALTER TABLE + Tabellename + ADD + Spaltenname + Datentyp + Attribut
 - Beispiel: ALTER TABLE mitglieder ADD mitglieder_nutzername VARCHAR(20) UNIQUE;
- Primärschlüssel hinzufügen (ADD PRIMARY KEY)
 - Befehl: ALTER TABLE + Tabellename + ADD PRIMARY KEY + (Spaltenname in Klammern)
 - Beispiel: ALTER TABLE mitglieder ADD PRIMARY KEY (mitglieder_id)
- Datentyp ändern (MODIFY)
 - Befehl: ALTER TABLE + Tabellename + MODIFY + Spaltenname + Datentyp
 - Beispiel: ALTER TABLE mitglieder MODIFY mitglieder_alter TINYINT UNSIGNED
- Bedingung NOT NULL hinzufügen (MODIFY)
 - Befehl: Wie davor + Datentyp inklusive NOT NULL
 - Beispiel: ALTER TABLE mitglieder MODIFY mitglieder_vorname VARCHAR(20) NOT NULL

ERSTE SCHRITTE: Datenbank anlegen und Tabellen erstellen

Nachträgliche Änderungen der Tabelle

- Spalte löschen (DROP)
 - Befehl: ALTER TABLE + Tabellenname + DROP + Spaltennamen
 - Beispiel: ALTER TABLE mitglieder DROP mitglieder_nutzername
- Tabelle oder Datenbank löschen
 - Tabelle löschen
 - DROP TABLE + Tabellenname
 - DROP TABLE mitglieder2;
 - Wichtig: Mit Vorsicht anwenden - löscht Inhalte unwiderruflich!
 - Datenbank löschen:
 - DROP DATABASE + Datenbankname
 - DROP DATABASE testdb1

WERTE EINFÜGEN, BEARBEITEN UND ABFRAGEN

- INSERT-Befehl: Werte einfügen
 - Einfügen eines kompletten Datensatzes:
 - Datensatz zur Tabelle mit dem Namen der Mitglieder hinzufügen:
 - `INSERT INTO mitglieder (mitglieder_vorname, mitglieder_name, mitglieder_telefon, mitglieder_alter) VALUES ("Franziska", "Heinze", 12345, 32);`
 - Spaltenamen: `mitglieder_vorname`, etc.
 - Werte nach Reihenfolge der Spalte: "Franziska", "Heinze", etc.
 - Einfügen von Datensätzen ohne Spaltennamen:
 - `INSERT INTO mitglieder VALUES (3, "Sebastian", "Becker", 2575577, 24);`
 - Die Werte müssen in Reihenfolge der Tabelle (Spalten) stehen

WERTE EINFÜGEN, BEARBEITEN UND ABFRAGEN

- SELECT-Befehl: Werte abfragen
 - Abrufen aller Daten aus der Tabelle
 - `SELECT * FROM mitglieder`
 - * steht für alle Spalten der Tabelle
 - Abrufen spezifischer Spalten
 - `SELECT mitglieder_name FROM mitglieder`
 - Abrufen mehrerer Spalten
 - `SELECT mitglieder_name, mitglieder_telefon FROM mitglieder`
 - Abrufen spezifischer Zeilen basierend auf dem Primärschlüssel (ID)
 - `SELECT mitglieder_name FROM mitglieder WHERE mitglieder_id = 2`
 - Abrufen aller Daten für eine spezifische Zeile basierend auf dem Primärschlüssel (ID)
 - `SELECT * FROM mitglieder WHERE mitglieder_id = 1`

WERTE EINFÜGEN, BEARBEITEN UND ABFRAGEN

- WHERE-Klausel: Bestimmung der Werte
 - SELECT mitglieder_telefon FROM mitglieder WHERE mitglieder_vorname = "Sebastian";
 - Die Telefonnummer von Zeile Sebastian rausfiltern
 - Mit der Bedingung = "Sebastian"
 - SELECT mitglieder_telefon FROM mitglieder WHERE mitglieder_vorname = "Franziska";
 - SELECT mitglieder_telefon, mitglieder_vorname FROM mitglieder WHERE mitglieder_alter < 28;
 - SELECT mitglieder_telefon, mitglieder_vorname FROM mitglieder WHERE mitglieder_alter <= 27;
 - SELECT mitglieder_telefon, mitglieder_vorname FROM mitglieder WHERE mitglieder_vorname <> "Patrick";
 - <> heißt ungleich

WERTE EINFÜGEN, BEARBEITEN UND ABFRAGEN

- UPDATE-Befehl: Werte ändern
 - Aktualisieren der Telefonnummer eines Mitglieds mit der ID 2
 - UPDATE mitglieder SET mitglieder_telefon = 8273646 WHERE mitglieder_id = 2;
 - Telefonnummer und Alter ändern
 - UPDATE mitglieder SET mitglieder_telefon = 8273646, mitglieder_alter = 42 WHERE mitglieder_id = 2;
- DELETE-Befehl: Werte löschen
 - DELETE FROM mitglieder WHERE mitglieder_id = 2;
 - Löschen eines Mitglieds mit der ID 2
 - DELETE FROM mitglieder WHERE mitglieder_alter > 50;
 - DELETE FROM mitglieder
 - ...

FREMDSCHLÜSSEL IN DER DATENBANK VERWENDEN

- DEFINITION

- Fremdschlüssel in der Datenbank
 - Formalisiert Beziehungen zwischen Tabellen
 - Normalfall: Bezieht sich auf Primärschlüssel einer anderen Tabelle
 - Erlaubt Modellierung von Beziehungen zwischen Datensätzen in verschiedenen Tabellen

FREMDSCHLÜSSEL EINFÜGEN BEIM ERSTELLEN DER TABELLE

- DEFINITION DES FREMDSCHLÜSSELS

- Befehl: FOREIGN KEY (Spalte) REFERENCES Referenz-Tabelle (Referenz-Spalte)
- Beispiel: FOREIGN KEY (sportangebote_teilnehmer) REFERENCES mitglieder (mitglieder_id)
- Einbau in den CREATE-TABLE-Befehl - Beispiel:

```
21 CREATE TABLE IF NOT EXISTS sportangebote2(
22 sportangebote_id INT PRIMARY KEY AUTO_INCREMENT,
23 sportangebote_sportart VARCHAR(20) NOT NULL,
24 sportangebote_teilnehmer INT NOT NULL,
25 FOREIGN KEY (sportangebote_teilnehmer) REFERENCES mitglieder (mitglieder_id)
26 );
```

- Daten einfügen durch INSERT-INTO-Befehl:

- INSERT INTO sportangebote2 (sportangebote_sportart, sportangebote_teilnehmer)
VALUES ("Volleyball", 1), ("Volleyball", 3), etc.

BEZIEHUNGEN ZWISCHEN TABELLEN MIT FREMD SCHLÜSSEL

- 1:1 Beziehungen: Erstellung der Tabelle mit 1:1 Beziehung

```
1 CREATE TABLE IF NOT EXISTS zugangsdaten(
2     zugangsdaten_id INT PRIMARY KEY AUTO_INCREMENT,
3     zugangsdaten_nutzername VARCHAR(20) NOT NULL,
4     zugangsdaten_passwort VARCHAR(20) NOT NULL,
5     zugangsdaten_mitglied INT NOT NULL UNIQUE,
6     FOREIGN KEY (zugangsdaten_mitglied) REFERENCES mitglieder(mitglieder_id)
7 );
```

- Beziehung hergestellt durch Fremdschlüssel "zugangsdaten_mitglied"
- Beschränkung auf 1:1 Beziehung durch Hinzufügen von UNIQUE zur Fremdschlüsselspalte:
 - zugangsdaten_mitglied INT NOT NULL UNIQUE;

BEZIEHUNGEN ZWISCHEN TABELLEN MIT FREMDSchLÜSSEL

- 1:n Beziehung: Erstellung der Tabelle 1:n Beziehung

```
14 CREATE TABLE IF NOT EXISTS sportveranstaltungen(
15     sportveranstaltungen_id INT PRIMARY KEY AUTO_INCREMENT,
16     sportveranstaltungen_sportart VARCHAR(20) NOT NULL,
17     sportveranstaltungen_mindestalter INT,
18     sportveranstaltungen_hoechstalter INT,
19     sportveranstaltungen_termin VARCHAR(40) NOT NULL,
20     sportveranstaltungen_trainer INT NOT NULL,
21     FOREIGN KEY (sportveranstaltungen_trainer) REFERENCES trainer(trainer_id)
22 );
```

- Im Gegensatz zu 1:1 Beziehung: UNIQUE nicht erforderlich:
 - Da ein Trainer mehrere Sportveranstaltungen leiten kann.
- Einfügen von Daten:
 - INSERT INTO sportveranstaltungen (sportveranstaltungen_sportart, sportveranstaltungen_mindestalter, sportveranstaltungen_hoechstalter, sportveranstaltungen_termin, sportveranstaltungen_trainer) VALUES ('Fußball', 6, 8, 'Mittwoch, 17:30 - 18:30', 6), ('Volleyball', 10, 14, 'Dienstag, 16:00 - 18:00', 3), ('Handball', 14, 35, 'Mittwoch, 19:30 - 21:00', 2), ('Fußball', 12, 14, 'Donnerstag, 18:00 - 19:30', 6);

BEZIEHUNGEN ZWISCHEN TABELLEN MIT FREMDSchLÜSSEL

- n:m Beziehungen - Erstellen der Tabelle für n:m Beziehung:
 - Für die Beziehungen zwischen sportveranstaltungen und mitglieder, wird eine neue Tabelle (sportangebote) erstellt. Dieser verweist auf die Tabellen sportveranstaltungen und mitglieder

```
25 CREATE TABLE IF NOT EXISTS sportangebote(
26     sportangebote_id INT PRIMARY KEY AUTO_INCREMENT,
27     sportangebote_veranstaltung INT,
28     sportangebote_mitglied INT,
29     FOREIGN KEY (sportangebote_veranstaltung) REFERENCES \
30     sportveranstaltungen(sportveranstaltungen_id),
31     FOREIGN KEY (sportangebote_mitglied) REFERENCES \
32     mitglieder(mitglieder_id)
33 );
```

- Die Tabelle hat zwei Fremdschlüssel - Damit verweist man auf sportveranstaltungen und mitglieder
- Die Tabelle dient als Bindeglied für n:m Beziehung

BEZIEHUNGEN ZWISCHEN TABELLEN MIT FREMDSchLÜSSEL

- n:m Beziehungen - Hinzufügen der Spalte für die Veranstaltung in der Tabelle sportangebote

- Sportangebote_veranstaltung in der Tabelle sportangebote hinzufügen:

```
ALTER TABLE sportangebote ADD sportangebote_veranstaltung INT;
```

- Deklaration der Fremdschlüsselbeziehung

Beziehung zwischen sportangebote_veranstaltung und sportveranstaltungen deklarieren

Hierdurch: Werte in sportangebote_veranstaltungen auf existierende Werte in der Tabelle sportveranstaltungen verwiesen.

```
ALTER TABLE sportangebote ADD FOREIGN KEY(sportangebote_veranstaltung) REFERENCES  
sportveranstaltungen(sportveranstaltungen_id);
```

BEZIEHUNGEN ZWISCHEN TABELLEN MIT FREMDSCHLÜSSEL

- n:m Beziehungen - Aktualisieren der Werte in der neuen Spalte:

UPDATE sportangebote SET sportangebote_veranstaltung = 3 WHERE sportangebote_id IN (8, 12);

UPDATE sportangebote SET sportangebote_veranstaltung = 6 WHERE sportangebote_id IN (1, 3, 15);

UPDATE sportangebote SET sportangebote_veranstaltung = 5 WHERE sportangebote_id IN (6, 14);

UPDATE sportangebote SET sportangebote_veranstaltung = 4 WHERE sportangebote_id = 11;

- Hier werden die Werte in die Spalte sportangebote_veranstaltung eingefügt.
- Jede Zeile in sportangebote verweist auf eine bestimmte Veranstaltung in sportveranstaltung.

PRAKTISCHE ANWENDUNG VON SQL - FALLBEISPIELE:

SQL wird in verschiedenen Branchen und Anwendungsgebieten weltweit eingesetzt, um auf strukturierte Daten zuzugreifen.

Unternehmensdatenbanken

- Kundeninformationen
- Bestellungen
- Lagerbestände und Finanzdaten

E-Commerce

- Produktinformationen
- Kundenprofile
- Bestellungen und Zahlungsdaten

Gesundheitswesen

- Patientenakten
- Medizinische Diagnosen
- Laborergebnisse und Abrechnungsdaten

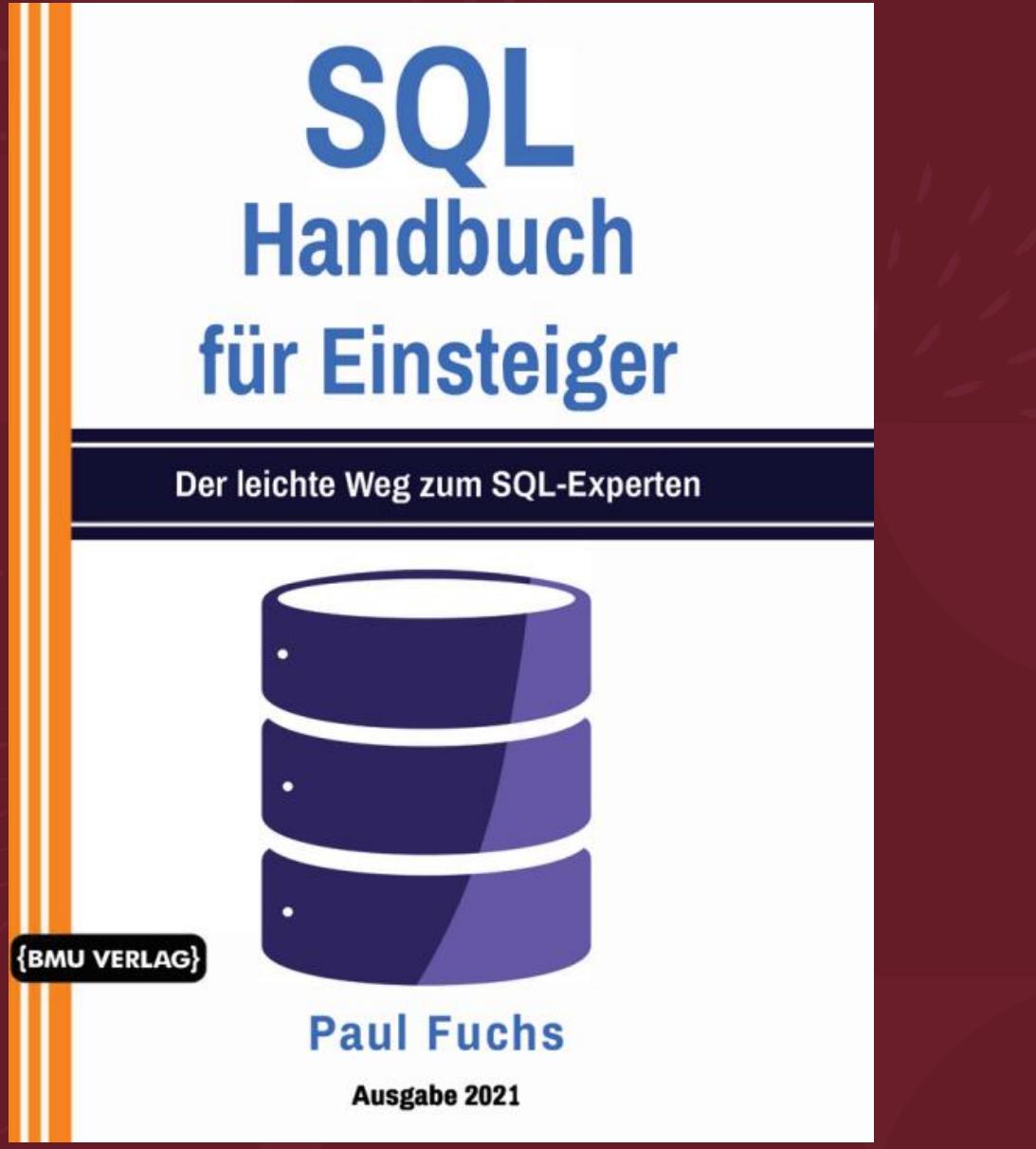
Bildungseinrichtungen

- Studentendaten
 - Kursinformationen
 - Prüfungsergebnisse
- ...
- ...

ZUSAMMENFASSUNG UND SCHLUSS

WICHTIGE PUNKTE DES VORTRAGS:

- Das relationale Datenbankmodell
 - Struktur für Datenablage, beeinflusst Zugriff auf Daten
 - Alternativen: Hierarchisch, Netzwerk, objektorientiert, objektrelationell
 - MySQL als beliebtes Datenbanksystem
- Datenbank anlegen und Tabellen erstellen
 - CREATE DATABASE, CREATE TABLE
 - Primärschlüssel, Bedingungen, Attribute
 - Änderungen an Tabelle: Hinzufügen, Löschen und Ändern
- Werte einfügen, bearbeiten und abfragen
 - INSERT, UPDATE, DELETE
 - SELECT: Abrufen von Daten, WHERE-Klausel für Bedingungen
- Fremdschlüssel in der Datenbank verwenden
 - Beziehungen zwischen Tabellen, FOREIGN KEY Beispiel mit Sportangeboten und Mitgliedern
- Durch SQL: Effiziente Datenverwaltung in verschiedenen Branchen



LITERATURVERZEICHNIS

- Fuchs, Paul (2021): SQL Handbuch für Einsteiger. Der leichte Weg zum SQL-Experten. (BMU Media Verlag).