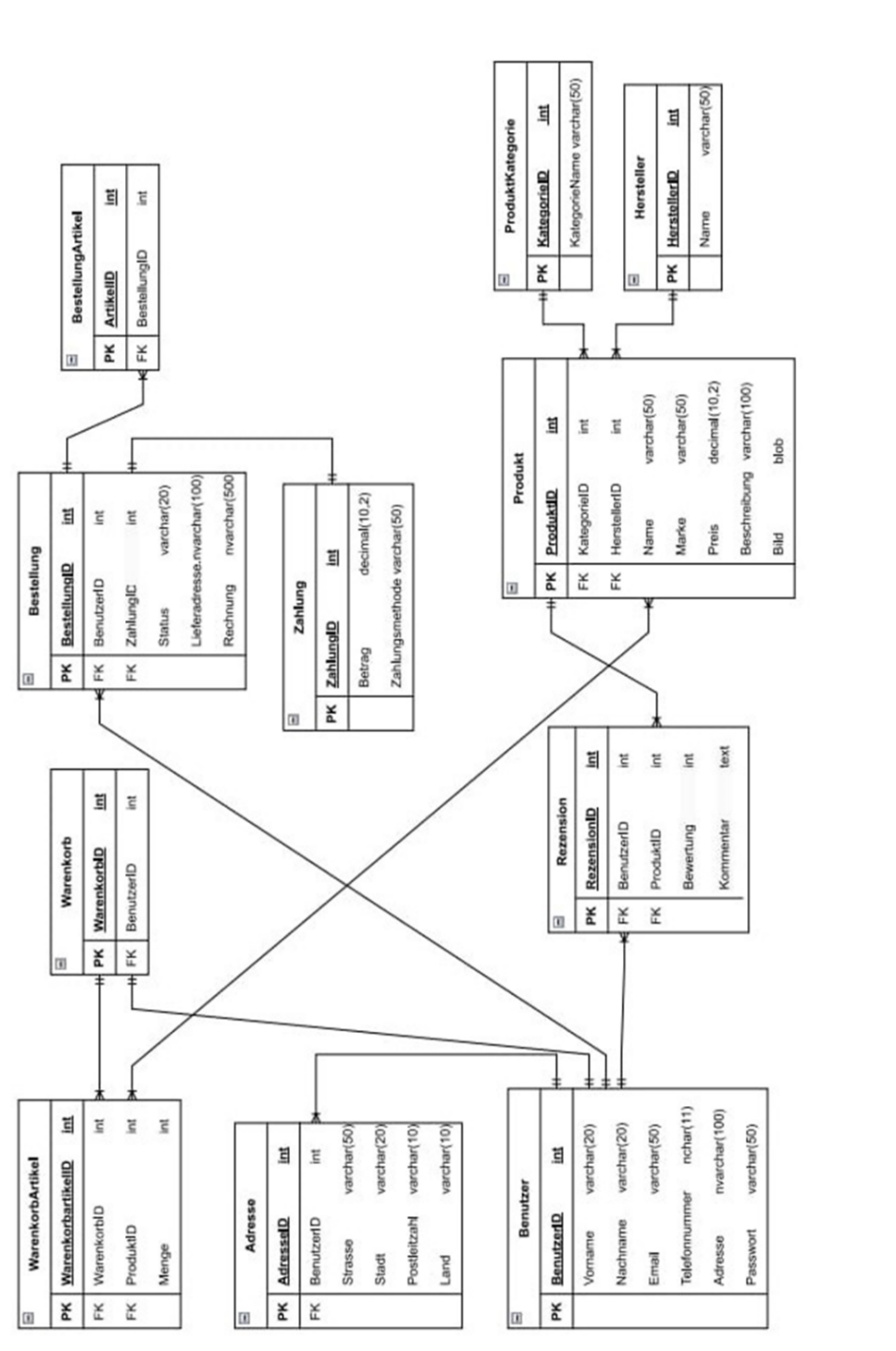
**DATENBANKEN**

**Doç. Dr. SELÇUK KIRAN**

**ER-Model-Project**

**E-Commerce Webseite von Campingausrüstung**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sümeyye Yetimoğlu**  **138621046** | **Şebnem Yüksel**  **199523086** | **Furkan Çelik**  **138419026** | **Ayşe Serra Toker**  **138620010** |
|  |  |  |  |



# CREATE TABLE TABELLEN

**-Nachfolgend finden Sie die Tabellen, die wir in MS SQL erstellt haben.**

* 1. CREATE TABLE WarenkorbArtikel(

WarenkorbArtikelID int,

WarenkorbID int,

ProduktID int,

Menge int,

CONSTRAINT pk\_WarenkorbArtikelId PRIMARY KEY (WarenkorbArtikelID)

)

* 1. CREATE TABLE Adresse(

AdresseID int,

BenutzerID int,

Strasse varchar(50),

Stadt varchar(20),

Postleitzahl varchar(10),

Land varchar(10),

CONSTRAINT pk\_AdresseID PRIMARY KEY (AdresseID)

)

* 1. CREATE TABLE Benutzer(

BenutzerID int PRIMARY KEY NOT NULL,

Vorname varchar(20),

Nachname varchar(20),

Email varchar(50) NOT NULL,

Telefonnummer nchar(11),

Adresse nvarchar(100),

Passwort varchar(50) NOT NULL

)

* 1. CREATE TABLE ProduktKategorie (

KategorieID INT PRIMARY KEY NOT NULL,

KategorieName VARCHAR(50) NOT NULL

)

* 1. CREATE TABLE Produkt(

ProduktID int PRIMARY KEY NOT NULL,

KategorieID int FOREIGN KEY REFERENCES ProduktKategorie(KategorieID),

HerstellerID int,

ProduktName varchar(50) NOT NULL,

Marke varchar(50),

Preis money,

Beschreibung varchar(100),

)

* 1. CREATE TABLE Rezension(

RezensionID int PRIMARY KEY NOT NULL,

BenutzerID int FOREIGN KEY REFERENCES Benutzer(BenutzerID) NOT NULL,

ProduktID int FOREIGN KEY REFERENCES Produkt(ProduktID) NOT NULL,

Bewertung int,

Kommentar varchar(250)

)

* 1. CREATE TABLE Zahlung(

ZahlungID int PRIMARY KEY NOT NULL,

Rechnung int NOT NULL,

Zahlungsmethode varchar(50)

)

* 1. CREATE TABLE Bestellung(

BestellungID int PRIMARY KEY NOT NULL,

BenutzerID int FOREIGN KEY REFERENCES Benutzer(BenutzerID) NOT NULL,

ZahlungID int NOT NULL,

Status varchar(100),

Lieferadresse nvarchar(100),

Rechnung nvarchar(500),

FOREIGN KEY (ZahlungID) REFERENCES Zahlung(ZahlungID)

)

* 1. CREATE TABLE Warenkorb (

WarenkorbID INT PRIMARY KEY NOT NULL,

BenutzerID INT NOT NULL,

FOREIGN KEY (BenutzerID) REFERENCES Benutzer(BenutzerID)

)

* 1. CREATE TABLE BestellungArtikel (

ArtikelID INT PRIMARY KEY NOT NULL,

BestellungID INT NOT NULL,

FOREIGN KEY (BestellungID) REFERENCES Bestellung(BestellungID)

)

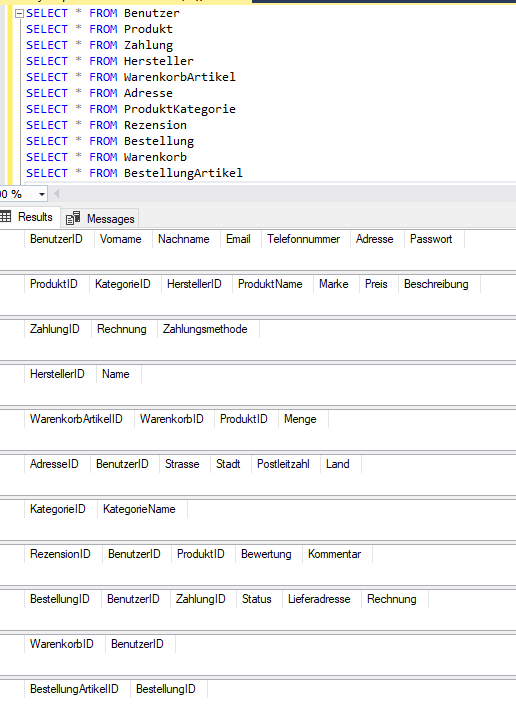
* 1. CREATE TABLE Hersteller (

HerstellerID INT PRIMARY KEY NOT NULL,

Name VARCHAR(50) NOT NULL

)

**-Die von uns in MS SQL erstellten Tabellen und Inhalte können Sie der folgenden Tabelle entnehmen.**



# 2. ALTER TABLE TABELLEN

**-Nachdem wir die Tabellen in MS SQL erstellt hatten, erstellten wir 5 Alter Tables.**

2.1. ALTER TABLE Benutzer

ADD Beschreibung varchar(225)

2.2. ALTER TABLE Produkt

ADD Farbe VARCHAR(30);

2.3. ALTER TABLE Bestellung

ADD BestellungVerify BIT;

2.4. ALTER TABLE Zahlung

ADD Transaktionsdatum DATETIME;

2.5. ALTER TABLE Hersteller

ADD Lagerbestand DECIMAL(10,2);

2.6. ALTER TABLE Produkt

ADD Stock smallint

**-Wir haben die von uns verkauften Produkte in 4 Kategorien unterteilt und jeder Kategorie eine ID zugewiesen**.

# INSERT INTO ANFRAGEN

**-Wir haben die von uns verkauften Produkte in 4 Kategorien unterteilt und jeder Kategorie eine ID zugewiesen**.

* 1. INSERT INTO ProduktKategorie ( KategorieID, KategorieName)

VALUES (100, 'Unterkunft')

* 1. INSERT INTO ProduktKategorie ( KategorieID, KategorieName)

VALUES(200 ,'Kochen')

* 1. INSERT INTO ProduktKategorie ( KategorieID, KategorieName)

VALUES(300, 'Beleuchtung')

* 1. INSERT INTO ProduktKategorie ( KategorieID, KategorieName)

VALUES(400, 'Sicherheit und Erste Hilfe')

**-Wir haben dem System einige Benutzer hinzugefügt.**

* 1. INSERT INTO Benutzer (BenutzerID, Vorname, Nachname, Email, Telefonnummer, Adresse, Passwort)

VALUES(1, 'Selçuk', 'Kıran', 'Selcukkiran@example.com', '542-4987', 'Göztepe MarmaraUniversität', 'kodadisql')

* 1. INSERT INTO Benutzer (BenutzerID, Vorname, Nachname, Email, Telefonnummer, Adresse, Passwort)

VALUES(2, 'Furkan', 'Çelik', 'furkancelikk@gmail.com', '556-9545', 'Beykoz Zerzavatçı Mahallesi', 'datenbankgecilecek')

* 1. INSERT INTO Benutzer (BenutzerID, Vorname, Nachname, Email, Telefonnummer, Adresse, Passwort)

VALUES(3, 'Max', 'Mustermann', 'max.mustermann@example.com', '689-2752', 'Musterstraße 1 12345 Musterstadt', '665433221')

* 1. INSERT INTO Benutzer (BenutzerID, Vorname, Nachname, Email, Telefonnummer, Adresse, Passwort)

VALUES(4, 'Alex', 'Jahx', 'alex-jahx12@gmail.com', '556-1694', 'Berlin', 'kals123')

* 1. INSERT INTO Benutzer (BenutzerID, Vorname, Nachname, Email, Telefonnummer, Adresse, Passwort)

VALUES(5, 'Brad', 'Pitt', 'bradpitt@gmail.com', '532-1223', 'New York', 'iamperfect12')

* 1. INSERT INTO Benutzer (BenutzerID, Vorname, Nachname, Email, Telefonnummer, Adresse, Passwort)

VALUES(6, 'Sandra', 'Flower', 'sandr-flower34@example.com', '577-9936', 'Mexico', 'beans123'

**- Wir haben 5 Hersteller hinzugefügt**

* 1. INSERT INTO Hersteller(HerstellerID,Name)

VALUES(4000,'Dasistok')

* 1. INSERT INTO Hersteller(HerstellerID,Name)

VALUES(4100,'Whatsup')

* 1. INSERT INTO Hersteller(HerstellerID,Name)

VALUES(4200,'Mountain')

* 1. INSERT INTO Hersteller(HerstellerID,Name)

VALUES(4300,'Tırpa')

* 1. INSERT INTO Hersteller(HerstellerID,Name)

VALUES(4400,'Walks')

**-Wir haben dem System einige Produkt hinzugefügt.**

* 1. INSERT INTO Produkt (ProduktID, KategorieID, HerstellerID, ProduktName, Marke, Preis, Beschreibung)

VALUES(1000, 100, 4000, 'Zelt', 'OutdoorPro', 199.99, 'Wasserdichtes und strapazierfähiges Zelt.')

* 1. INSERT INTO Produkt (ProduktID, KategorieID, HerstellerID, ProduktName, Marke, Preis, Beschreibung)

VALUES(1010, 100, 4000, 'Schlafsack', 'SleepWell', 79.99, 'Warmer und bequemer Schlafsack.')

* 1. INSERT INTO Produkt (ProduktID, KategorieID, HerstellerID, ProduktName, Marke, Preis, Beschreibung)

VALUES(1020, 100, 4000, 'Matte', 'TrailBlaze', 29.99, 'Leichte und tragbare Campingmatte.')

* 1. INSERT INTO Produkt (ProduktID, KategorieID, HerstellerID, ProduktName, Marke, Preis, Beschreibung)

VALUES(1030, 100, 4000, 'Hängematte', 'RelaxZone', 49.99, 'Ideale Hängematte zum Entspannen.')

* 1. INSERT INTO Produkt (ProduktID, KategorieID, HerstellerID, ProduktName, Marke, Preis, Beschreibung)

VALUES(1040, 100, 4000, 'Insektenschutz', 'BugGuard', 19.99, 'Insektenschutz, der Insekten fernhält.')

* 1. INSERT INTO Produkt (ProduktID, KategorieID, HerstellerID, ProduktName, Marke, Preis, Beschreibung)

VALUES(1050, 200, 4100, 'Campingkocher', 'FireMaster', 149.99, 'Tragbare und leistungsstarke Campingkocher.')

* 1. INSERT INTO Produkt (ProduktID, KategorieID, HerstellerID, ProduktName, Marke, Preis, Beschreibung)

VALUES(1060, 200, 4100, 'Campingtopfset', 'CookMate', 79.99, 'Leichtes und strapazierfähiges Campingtopfset.')

* 1. INSERT INTO Produkt (ProduktID, KategorieID, HerstellerID, ProduktName, Marke, Preis, Beschreibung)

VALUES(1070, 200, 4100, 'Tragbarer Campingofen', 'HeatMax', 129.99, 'Tragbarer und kompakter Campingofen.')

* 1. INSERT INTO Produkt (ProduktID, KategorieID, HerstellerID, ProduktName, Marke, Preis, Beschreibung)

VALUES(1080, 200, 4200, 'Grill', 'GrillMaster', 199.99, 'Ideal für Grillen und BBQ.')

* 1. INSERT INTO Produkt (ProduktID, KategorieID, HerstellerID, ProduktName, Marke, Preis, Beschreibung)

VALUES(1090, 200, 4200, 'Campinggeschirrset', 'DineInStyle', 49.99, 'Leichtes und tragbares Campinggeschirrset.')

* 1. INSERT INTO Produkt (ProduktID, KategorieID, HerstellerID, ProduktName, Marke, Preis, Beschreibung)

VALUES(1100, 200, 4300, 'Wärmflasche', 'HotSip', 24.99, 'Tragbare Wärmflasche für heiße Getränke.')

* 1. INSERT INTO Produkt (ProduktID, KategorieID, HerstellerID, ProduktName, Marke, Preis, Beschreibung)

VALUES(1120, 300, 4300, 'Stirnlampe', 'HeadLightPro', 29.99, 'Leistungsstarke Stirnlampe für freihändige Nutzung.')

* 1. INSERT INTO Produkt (ProduktID, KategorieID, HerstellerID, ProduktName, Marke, Preis, Beschreibung)

VALUES(1130, 300, 4400, 'Campinglampen', 'LightUpLife', 49.99, 'Tragbare Lampen zur Beleuchtung des Campingplatzes.')

* 1. INSERT INTO Produkt (ProduktID, KategorieID, HerstellerID, ProduktName, Marke, Preis, Beschreibung)

VALUES(1140, 300, 4400, 'Karten- und Kompassset', 'ExploreMate', 39.99, 'Karten- und Kompassset für die richtige Orientierung.')

**-Wir haben die Adressen der Benutzer hinzugefügt**

* 1. INSERT INTO Adresse(AdresseID,BenutzerID,Strasse,Stadt,Postleitzahl,Land)

VALUES(5010,1,'eğitim','İstanbul','34722','Turkey')

* 1. INSERT INTO Adresse(AdresseID,BenutzerID,Strasse,Stadt,Postleitzahl,Land)

VALUES(5020,2,'zerzavatçı','İstanbul','34752','Turkey')

* 1. INSERT INTO Adresse(AdresseID,BenutzerID,Strasse,Stadt,Postleitzahl,Land)

VALUES(5030,3,'Musterstrasse','Berlin','25612','German')

* 1. INSERT INTO Adresse(AdresseID,BenutzerID,Strasse,Stadt,Postleitzahl,Land)

VALUES(5040,4,'klein','berlin','45689','German')

* 1. INSERT INTO Adresse(AdresseID,BenutzerID,Strasse,Stadt,Postleitzahl,Land)

VALUES(5050,5,'Highway','New york','56489','USA')

* 1. INSERT INTO Adresse(AdresseID,BenutzerID,Strasse,Stadt,Postleitzahl,Land)

VALUES(5060,6,'Taxo','Mexico','68952','Mexico')

**-Wir haben die notwendigen Felder in der Tabelle warenkorbartikel ausgefüllt.**

* 1. INSERT INTO WarenkorbArtikel(WarenkorbArtikelID,WarenkorbID,ProduktID,Menge)

VALUES (6000,12000,1000,2)

* 1. INSERT INTO WarenkorbArtikel(WarenkorbArtikelID,WarenkorbID,ProduktID,Menge)

VALUES(6010,12010,1060,1)

* 1. INSERT INTO WarenkorbArtikel(WarenkorbArtikelID,WarenkorbID,ProduktID,Menge)

VALUES(6020,12020,1040,1)

* 1. INSERT INTO WarenkorbArtikel(WarenkorbArtikelID,WarenkorbID,ProduktID,Menge)

VALUES(6030,12030,1030,2)

**-Wir haben den Warenkorbtisch so gefüllt, wie er sein sollte.**

|  |  |
| --- | --- |
| * 1. INSERT INTO Warenkorb   VALUES(12000,2)   * 1. INSERT INTO Warenkorb   VALUES(12010,10)   * 1. INSERT INTO Warenkorb   VALUES(12020,8)   * 1. INSERT INTO Warenkorb   VALUES(12030,3)   * 1. INSERT INTO Warenkorb   VALUES(12040,5) |  |

**-Wir haben den Zahlungstisch so gefüllt, wie er sein sollte.**

* 1. INSERT INTO Zahlung

VALUES(7000,399.98,'Kredite')

* 1. INSERT INTO Zahlung

VALUES(7010,34.99,'Kredite')

* 1. INSERT INTO Zahlung

VALUES(7020,19.99,'Kredite')

* 1. INSERT INTO Zahlung

VALUES(7030,49.99,'Kredite')

* 1. INSERT INTO Zahlung

VALUES(7040,69.99,'Kredite')

**-Wir haben die Bestelltabelle unter Berücksichtigung aller von uns ausgefüllten Tabellen ausgefüllt.**

* 1. INSERT INTO Bestellung(BestellungID,BenutzerID,ZahlungID,Status,Lieferadresse,Rechnung)

VALUES(20000,2,7000,'Wurde geliefert','İstanbul',399)

* 1. INSERT INTO Bestellung(BestellungID,BenutzerID,ZahlungID,Status,Lieferadresse,Rechnung)

VALUES(20010,6,7010,'Wurde geliefert','Paris',34)

* 1. INSERT INTO Bestellung(BestellungID,BenutzerID,ZahlungID,Status,Lieferadresse,Rechnung)

VALUES(20020,4,7020,'Wurde geliefert','Trabzon',19)

* 1. INSERT INTO Bestellung(BestellungID,BenutzerID,ZahlungID,Status,Lieferadresse,Rechnung)

VALUES(20030,3,7030,'Wurde geliefert','Berlin',49)

* 1. INSERT INTO Bestellung(BestellungID,BenutzerID,ZahlungID,Status,Lieferadresse,Rechnung)

VALUES(20040,5,7040,'Wurde geliefert','New York',69)

**-Es wurden Kommentare zu unseren Produkten abgegeben.**

* 1. INSERT INTO Rezension (RezensionID, BenutzerID, ProduktID, Bewertung, Kommentar)

VALUES (1, 1, 1000, 5, 'Sehr gute Qualität!')

* 1. INSERT INTO Rezension (RezensionID, BenutzerID, ProduktID, Bewertung, Kommentar)

VALUES (2, 4, 1000, 5, 'Perfekt für ein Kurztrip.')

* 1. INSERT INTO Rezension (RezensionID, BenutzerID, ProduktID, Bewertung, Kommentar)

VALUES (3, 2, 1010, 2, 'Es gibt bessere Produkte auf dem Markt.')

* 1. INSERT INTO Rezension (RezensionID, BenutzerID, ProduktID, Bewertung, Kommentar)

VALUES (4, 1, 1040, 4, 'Hat alle Insekten ferngehalten, top!')

* 1. INSERT INTO Rezension (RezensionID, BenutzerID, ProduktID, Bewertung, Kommentar)

VALUES (5, 3, 1030, 1, 'Die Kinder hatten viel Spaß beim Abhängen ;)')

# DROP TABLE BEFEHLEN

**-Wir haben zwei verschiedene Tabellen für die beiden gewünschten Drop-Tabellen erstellt und den Drop durchgeführt.**

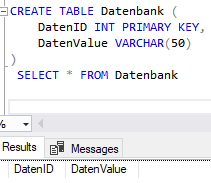
* 1. CREATE TABLE Datenbank (

DatenID INT PRIMARY KEY,

DatenValue VARCHAR(50)

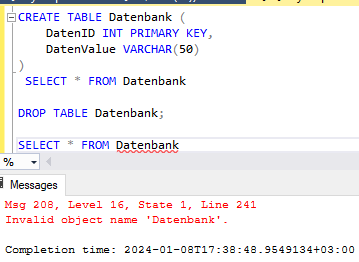
)

SELECT \* FROM Datenbank;



DROP TABLE Datenbank;

SELECT \* FROM Datenbank;



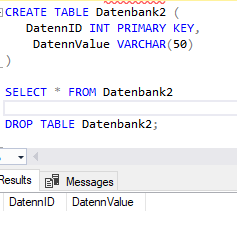
* 1. CREATE TABLE Datenbank2 (

DatennID INT PRIMARY KEY,

DatennValue VARCHAR(50)

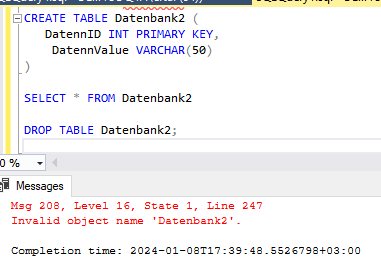
)

SELECT \* FROM Datenbank2



DROP TABLE Datenbank2;

SELECT \* FROM Datenbank2



# VIEW TABELLEN

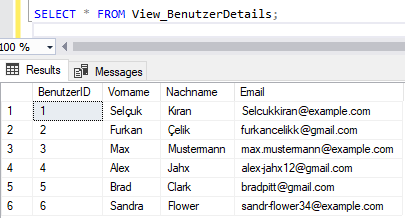
**-Wir haben 10 verschiedene Create View-Tabellen erstellt.**

* 1. CREATE VIEW View\_BenutzerDetails AS

SELECT BenutzerID, Vorname, Nachname, Email

FROM Benutzer;

SELECT \* FROM View\_BenutzerDetails;



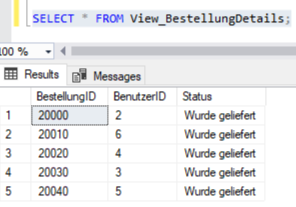
**Wir haben eine View-Abfrage erstellt, in der wir die IDs, E-Mail-Adressen, Namen und Nachnamen der Benutzer anzeigen.**

* 1. CREATE VIEW View\_BestellungDetails AS

SELECT BestellungID, BenutzerID, Status

FROM Bestellung;

SELECT \* FROM View\_BestellungDetails;



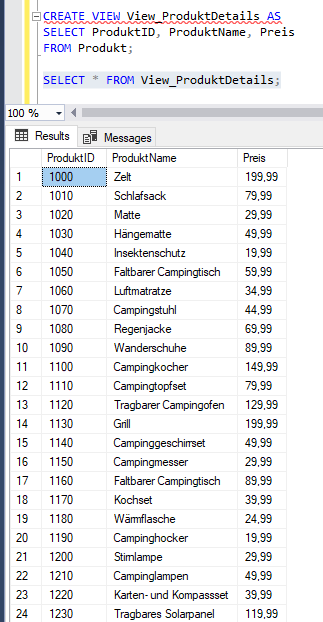
**Wir haben die Benutzer- und Bestellungtabellen kombiniert, um die Auftragsdetails zu sehen.**

* 1. CREATE VIEW View\_ProduktDetails AS

SELECT ProduktID, ProduktName, Preis

FROM Produkt;

SELECT \* FROM View\_ProduktDetails;



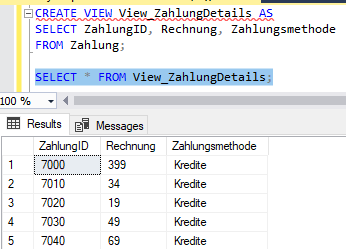
**Um die Produktdetails zu sehen, haben wir die Spalten produktid, produktname und preis aus der Tabelle produkt aufgerufen.**

* 1. CREATE VIEW View\_ZahlungDetails AS

SELECT ZahlungID, Betrag, Zahlungsmethode

FROM Zahlung;

SELECT \* FROM View\_ZahlungDetails;



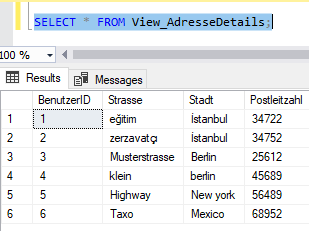
**Wir haben die Parameter zahlungid, rechnung und zahlungsmethode aus der Tabelle zahlung aufgerufen, um die Zahlungsdaten zu erhalten.**

* 1. CREATE VIEW View\_AdresseDetails AS

SELECT BenutzerID, Strasse, Stadt, Postleitzahl

FROM Adresse;

SELECT \* FROM View\_AdresseDetails;



**Um die Adressdetails aus der Adresstabelle zu sehen, haben wir die Parameter benutzerid, stresse, stadt und postleitzahl aufgerufen und visualisiert.**

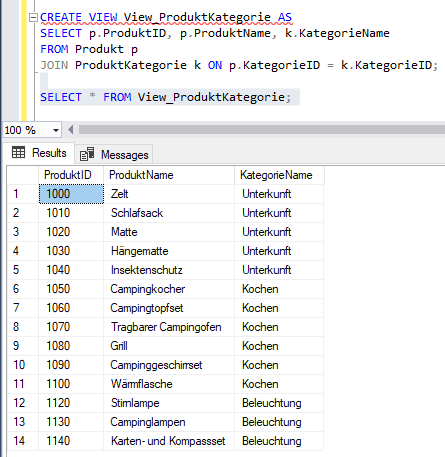
* 1. CREATE VIEW View\_ProduktKategorie AS

SELECT p.ProduktID, p.ProduktName, k.KategorieName

FROM Produkt p

JOIN ProduktKategorie k ON p.KategorieID = k.KategorieID;

SELECT \* FROM View\_ProduktKategorie;



**Um die Produktkategorien zu sehen, haben wir die Tabellen produkt und kategorie mit dem Befehl join verbunden und visualisiert**

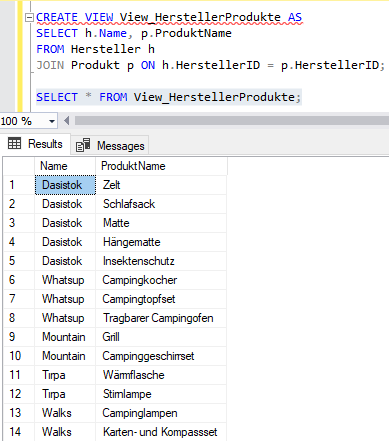
* 1. CREATE VIEW View\_HerstellerProdukte AS

SELECT h.Name, p.ProduktName

FROM Hersteller h

JOIN Produkt p ON h.HerstellerID = p.HerstellerID;

SELECT \* FROM View\_HerstellerProdukte;



**Um herauszufinden, wer die Lieferanten sind und welche Produkte sie verkaufen, haben wir zwei Tabellen mit dem Befehl join verbunden und visualisiert.**

* 1. CREATE VIEW View\_BestellungZahlung AS

SELECT bo.BestellungID, bo.Status, z.Betrag

FROM Bestellung bo

JOIN Zahlung z ON bo.ZahlungID = z.ZahlungID;

SELECT \* FROM View\_BestellungZahlung;

metin, elektronik donanım, ekran görüntüsü, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Um die Bestellpreise zu sehen, haben wir die Tabellen Bestellung und Zahlung mit dem Befehl join verbunden und visualisiert.**

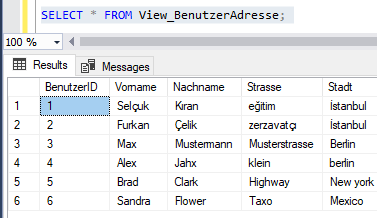
* 1. CREATE VIEW View\_BenutzerAdresse AS

SELECT b.BenutzerID, b.Vorname, b.Nachname, a.Strasse, a.Stadt

FROM Benutzer b

JOIN Adresse a ON b.BenutzerID = a.BenutzerID;

SELECT \* FROM View\_BenutzerAdresse;



**Um die Benutzeradressen zu sehen, haben wir die Benutzer- und Adresstabelle mit dem Befehl join verbunden und visualisiert.**

**-Wir haben 5 logische Indizes erstellt**

# INDEX TABELLEN

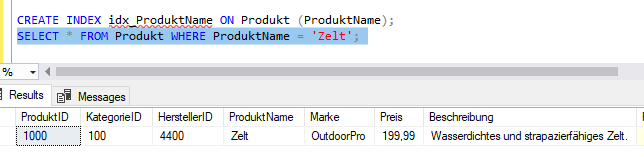
**-Wir haben 5 logische Indizes erstellt**

* 1. CREATE INDEX idx\_Email ON Benutzer (Email);

SELECT \* FROM Benutzer WHERE Email = 'example@example.com';

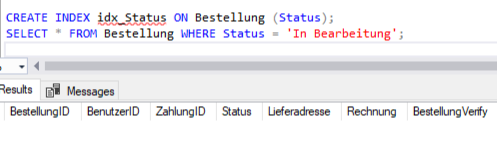
* 1. CREATE INDEX idx\_ProduktName ON Produkt (ProduktName);

SELECT \* FROM Produkt WHERE ProduktName = 'Zelt';



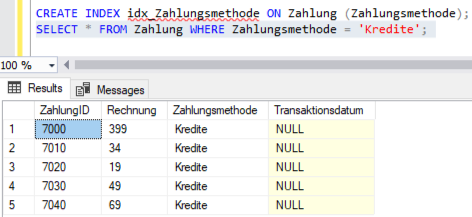
* 1. CREATE INDEX idx\_Status ON Bestellung (Status);

SELECT \* FROM Bestellung WHERE Status = 'In Bearbeitung';



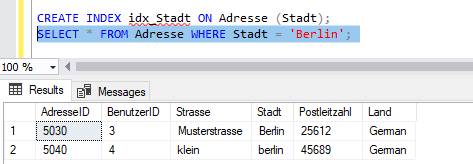
* 1. CREATE INDEX idx\_Zahlungsmethode ON Zahlung (Zahlungsmethode);

SELECT \* FROM Zahlung WHERE Zahlungsmethode = 'Kredite';



* 1. CREATE INDEX idx\_Stadt ON Adresse (Stadt);

SELECT \* FROM Adresse WHERE Stadt = 'Berlin';



# UPDATE TABELLEN

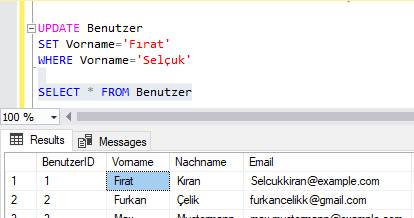
**-Wir haben 5 verschiedene Update-Datensätze erstellt.**

* 1. UPDATE Benutzer

SET Vorname='Fırat'

WHERE Vorname='Selçuk'

SELECT \* FROM Benutzer



* 1. UPDATE Produkt

SET HerstellerID=4400

WHERE HerstellerID=4000

SELECT \* FROM Produkt

metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

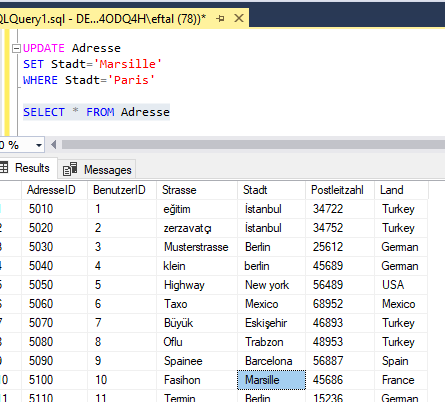
Açıklama otomatik olarak oluşturuldu

* 1. UPDATE Adresse

SET Stadt='Marsille'

WHERE Stadt='Paris'

SELECT \* FROM Adresse

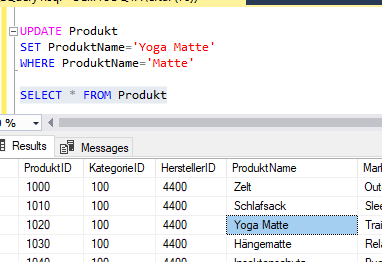


* 1. UPDATE Produkt

SET ProduktName='Yoga Matte'

WHERE ProduktName='Matte'

SELECT \* FROM Produkt

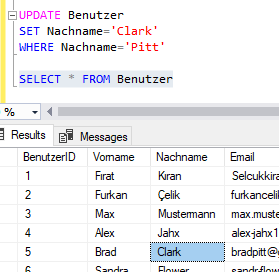


* 1. UPDATE Benutzer

SET Nachname='Clark'

WHERE Nachname='Pitt'

SELECT \* FROM Benutzer



# TRANSACTION OPERATIONEN

**-Transaktionen sind Operationen, die Erfolg und Fehler in Datenbankoperationen garantieren**

BEGIN TRANSACTION;

BEGIN TRY

INSERT INTO Bestellung (BestellungID, BenutzerID, Status, Lieferadresse, Rechnung)

VALUES (20000, 1, 'In Bearbeitung', 'Istanbul', 399)

INSERT INTO Zahlung (ZahlungID, Rechnung, Zahlungsmethode)

VALUES (7000, 399, 'Kredite');

COMMIT TRANSACTION;

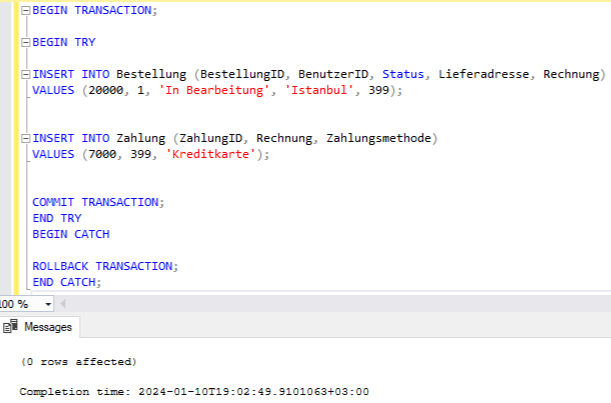
END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

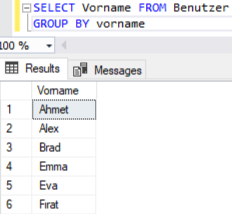
END CATCH;

**Beschreibung: Dieses SQL-Skript wurde geschrieben, um zwei INSERT-Operationen in einer Datenbank durchzuführen. Der Transaktionsblock ist in eine TRANSACTION mit einer TRY...CATCH-Struktur zur Fehlerbehandlung eingeschlossen. Die erste INSERT-Anweisung fügt einen neuen Datensatz in die Tabelle Bestellung ein und die zweite INSERT-Anweisung fügt einen Datensatz in die Tabelle Zahlung ein. Wenn beide INSERT-Anweisungen erfolgreich abgeschlossen wurden, soll die Anweisung COMMIT TRANSACTION; die Transaktionen dauerhaft in der Datenbank festschreiben. Wenn ein Fehler auftritt, rollt die Anweisung ROLLBACK TRANSACTION; im CATCH-Block alle Transaktionen zurück, so dass die Konsistenz der Datenbank erhalten bleibt.**

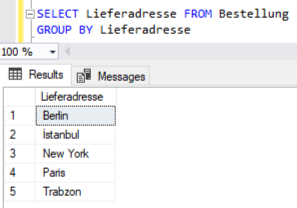


# SELECT BEFEHLEN

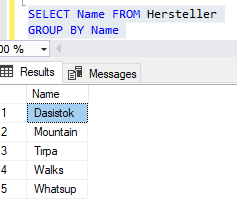
* 1. SELECT Vorname FROM Benutzer GROUP BY vorname



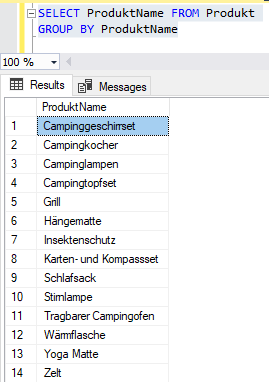
* 1. SELECT Lieferadresse FROM Bestellung GROUP BY Lieferadresse



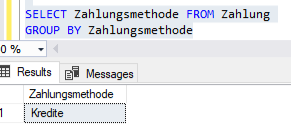
* 1. SELECT Name FROM Hersteller GROUP BY Name



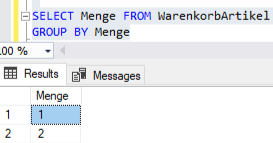
* 1. SELECT ProduktName FROM Produkt GROUP BY ProduktName



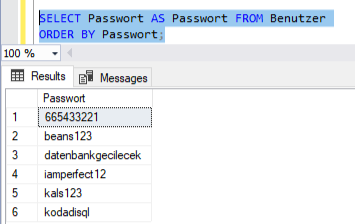
* 1. SELECT Zahlungsmethode FROM Zahlung GROUP BY Zahlungsmethode



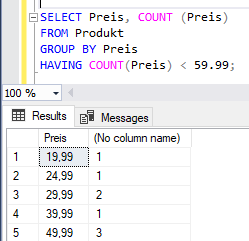
* 1. SELECT Menge FROM WarenkorbArtikel GROUP BY Menge



* 1. SELECT Passwort AS Passwort FROM Benutzer ORDER BY Passwort;

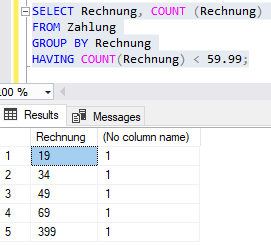


* 1. SELECT Preis, COUNT (Preis) FROM Produkt GROUP BY Preis HAVING COUNT(Preis) < 59.99;

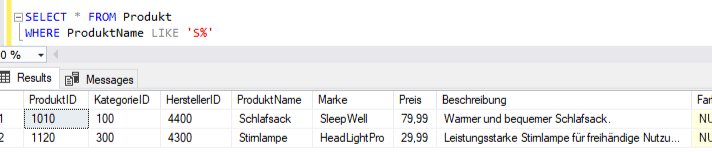


* 1. SELECT Rechnung, COUNT (Rechnung) FROM Zahlung GROUP BY Rechnung

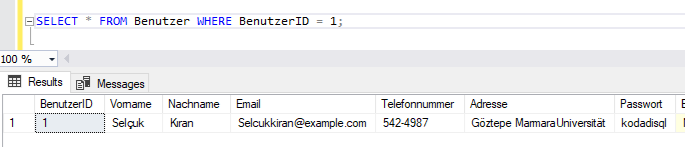
HAVING COUNT(Rechnung) < 59.99;



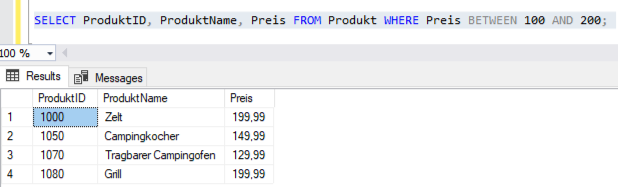
* 1. SELECT \* FROM Produkt WHERE ProduktName LIKE 'S%'



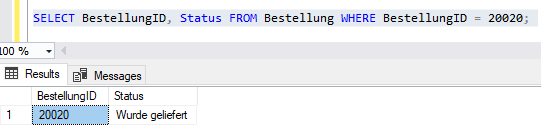
* 1. SELECT \* FROM Benutzer WHERE BenutzerID = 1;



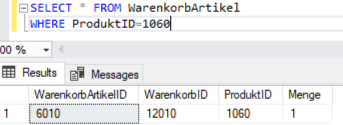
* 1. SELECT ProduktID, ProduktName, Preis FROM Produkt WHERE Preis BETWEEN 100 AND 200;



* 1. SELECT BestellungID, Status FROM Bestellung WHERE BestellungID = 20020;



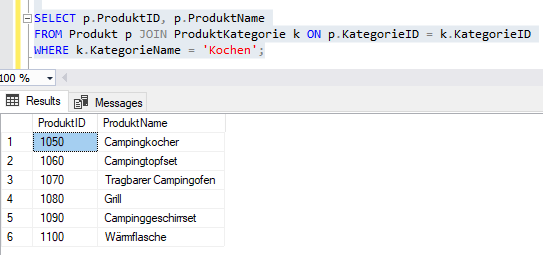
* 1. SELECT \* FROM WarenkorbArtikel WHERE ProduktID=1060



* 1. SELECT p.ProduktID, p.ProduktName

FROM Produkt p JOIN ProduktKategorie k ON p.KategorieID = k.KategorieID

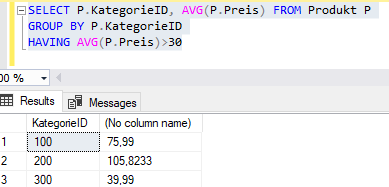
WHERE k.KategorieName = 'Kochen';



* 1. SELECT P.KategorieID, AVG(P.Preis) FROM Produkt P

GROUP BY P.KategorieID

HAVING AVG(P.Preis)>30

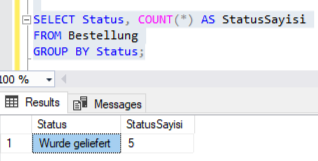


* 1. SELECT KategorieID, COUNT(\*) AS ProduktAnzahl FROM Produkt GROUP BY KategorieID;

metin, ekran görüntüsü, yazı tipi, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

* 1. SELECT Status, COUNT(\*) AS StatusSayisi FROM Bestellung GROUP BY Status;



* 1. SELECT \* FROM WarenkorbArtikel

WHERE ProduktID=1060

metin, ekran görüntüsü, ekran, görüntüleme, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

# JOIN BEFEHLEN

* 1. SELECT ProduktName, P.HerstellerID FROM Produkt P

INNER JOIN Hersteller H

ON P.HerstellerID= H.HerstellerID

* 1. SELECT Vorname, NachName FROM Benutzer B

INNER JOIN Warenkorb W

ON B.BenutzerID=W.BenutzerID

metin, ekran görüntüsü, ekran, görüntüleme, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

* 1. SELECT h.Name, COUNT(p.ProduktID) AS Anzahl\_der\_verkauften\_Produkte

FROM Hersteller h

INNER JOIN Produkt p ON h.HerstellerID = p.HerstellerID

GROUP BY h.Name

metin, ekran görüntüsü, ekran, görüntüleme, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

* 1. SELECT ProduktName, P.HerstellerID FROM Produkt P

INNER JOIN Hersteller H

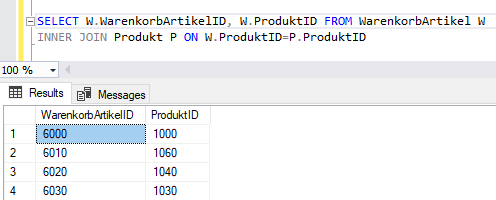
ON P.HerstellerID= H.HerstellerID

metin, ekran görüntüsü, sayı, numara, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

* 1. SELECT W.WarenkorbArtikelID, W.ProduktID FROM WarenkorbArtikel W

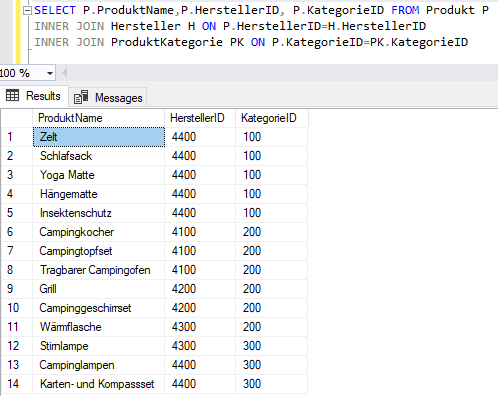
INNER JOIN Produkt P ON W.ProduktID=P.ProduktID



* 1. SELECT P.ProduktName,P.HerstellerID, P.KategorieID FROM Produkt P

INNER JOIN Hersteller H ON P.HerstellerID=H.HerstellerID

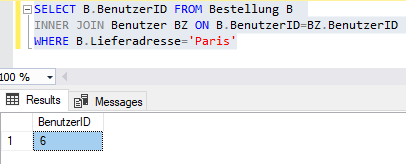
INNER JOIN ProduktKategorie PK ON P.KategorieID=PK.KategorieID



* 1. SELECT B.BenutzerID FROM Bestellung B

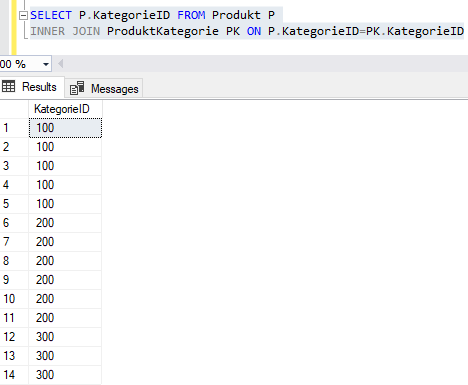
INNER JOIN Benutzer BZ ON B.BenutzerID=BZ.BenutzerID

WHERE B.Lieferadresse='Paris'



* 1. SELECT P.KategorieID FROM Produkt P

INNER JOIN ProduktKategorie PK ON P.KategorieID=PK.KategorieID

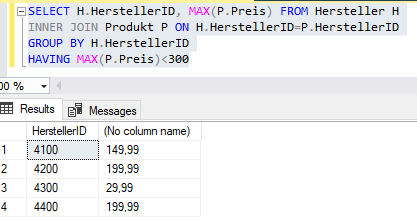


* 1. SELECT H.HerstellerID, MAX(P.Preis) FROM Hersteller H

INNER JOIN Produkt P ON H.HerstellerID=P.HerstellerID

GROUP BY H.HerstellerID

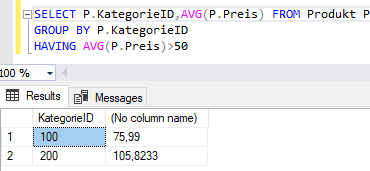
HAVING MAX(P.Preis)<300



* 1. SELECT P.KategorieID,AVG(P.Preis) FROM Produkt P

GROUP BY P.KategorieID

HAVING AVG(P.Preis)>50

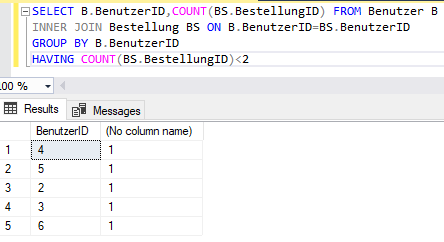


* 1. SELECT B.BenutzerID,COUNT(BS.BestellungID) FROM Benutzer B

INNER JOIN Bestellung BS ON B.BenutzerID=BS.BenutzerID

GROUP BY B.BenutzerID

HAVING COUNT(BS.BestellungID)<2



* 1. SELECT R.ProduktID, R.Kommentar FROM Rezension R

INNER JOIN Produkt P

ON P.ProduktID=R. ProduktID

SELECT \* FROM Rezension

metin, ekran görüntüsü, ekran, görüntüleme, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

* 1. SELECT R.ProduktID, R.Kommentar FROM Rezension R

INNER JOIN Produkt P

ON P.ProduktID=R. ProduktID

SELECT \* FROM Rezension

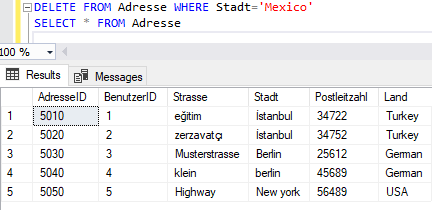
metin, ekran görüntüsü, ekran, görüntüleme, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

# DELETE BEFEHLEN

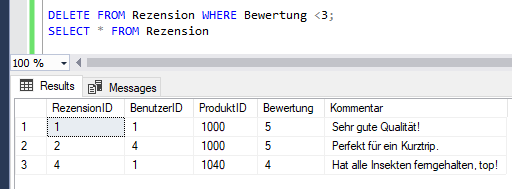
* 1. DELETE FROM Adresse WHERE Stadt='Mexico'

SELECT \* FROM Adresse



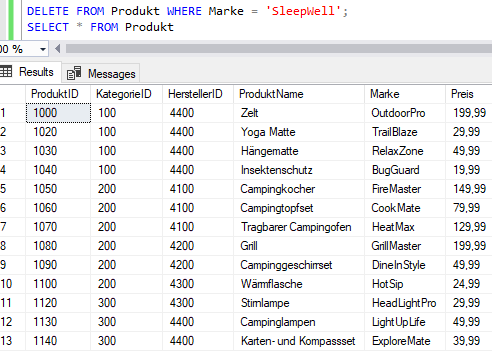
* 1. DELETE FROM Rezension WHERE Bewertung <3;

SELECT \* FROM Rezension



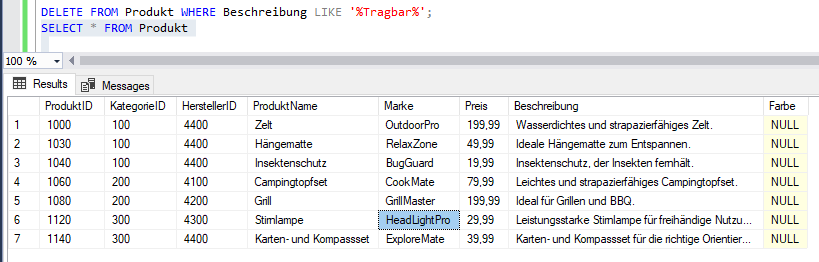
* 1. DELETE FROM Produkt WHERE Marke = 'SleepWell';

SELECT \* FROM Produkt



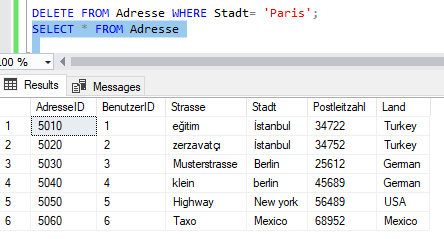
* 1. DELETE FROM Produkt WHERE Beschreibung LIKE '%Tragbar%';

SELECT \* FROM Produkt



* 1. DELETE FROM Adresse WHERE Stadt= 'Paris';

SELECT \* FROM Adresse



# CREATE PROCEDURE BEFEHLEN

* 1. CREATE PROCEDURE sp\_Adresse

AS

BEGIN

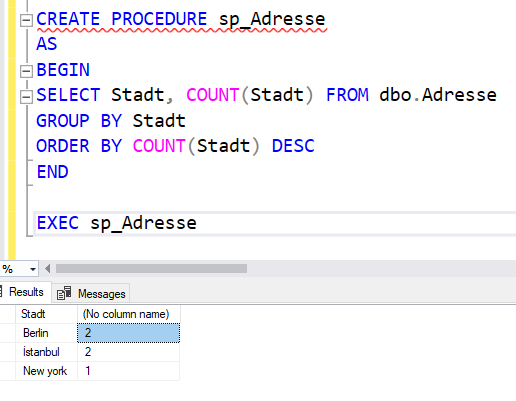
SELECT Stadt, COUNT(Stadt) FROM dbo.Adresse

GROUP BY Stadt

ORDER BY COUNT(Stadt) DESC

END

EXEC sp\_Adresse



**Beschreibung: Wir erstellen eine Prozedur namens sp\_Adresse und führen sie aus. Der Inhalt der gespeicherten Prozedur ist, dass wir die Städte und die Anzahl der Adressen in jeder Stadt aus der Spalte Stadt in der Tabelle dbo.Adresse ausgewählt haben. Anschließend werden diese Informationen nach Städten gruppiert und in absteigender Reihenfolge nach der Anzahl der Adressen sortiert. EXEC sp\_Adresse führt diese Prozedur aus, um eine Liste mit der Anzahl der Adressen in jeder Stadt zu erhalten, sortiert vom höchsten zum niedrigsten Wert.**

# CREATE TRIGGER BEFEHLEN

* 1. CREATE TRIGGER Produkt\_zahlung

ON Zahlung

AFTER INSERT

AS

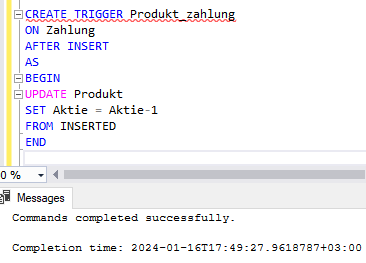
BEGIN

UPDATE Produkt

SET Aktie = Aktie-1

FROM INSERTED

END



**Beschreibung: Dieser SQL-Befehl erstellt einen Trigger, Produkt\_zahlung, der ausgelöst wird, wenn eine Zeile in die Tabelle Zahlung eingefügt wird. Wenn der Trigger aktiviert ist, d.h. wenn eine INSERT-Operation in der Zahlungstabelle durchgeführt wird, führt er einen UPDATE-Befehl aus, der die Produkttabelle aktualisiert. Der UPDATE-Befehl setzt die Spalte "Aktie" in der Tabelle "Produkt" um 1 von ihrem aktuellen Wert herab. INSERTED ist hier eine spezielle Tabelle und stellt die Zeilen dar, die während der INSERT-Operation hinzugefügt wurden. Wir haben diesen Trigger erstellt, um den Bestand verfolgen zu können.**