COUNT() zähle die gewählten Zeilen.

SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country  
ORDER BY COUNT(CustomerID) DESC;

AVG() Durchschnitt einer Zeile mit numerischen Werten.

SUM() Summe einer Zeile mit numerischen Werten.

MIN() MAX() Kleinste / Größtes Element der gewählten Zeile.

* **(INNER) JOIN:** Alle Werte, die in beiden Tabellen sind
* **LEFT (OUTER) JOIN:** Alle Werte der linken Tabelle und alle passenden Werte der rechten Tabelle
* **RIGHT (OUTER) JOIN:** Alle Werte der rechten Tabelle und alle passenden Werte der linken Tabelle
* **FULL (OUTER) JOIN:** Alle Wert, die in der rechten oder der linken Tabelle sind

Geben Sie für alle AI-Studierenden aus, ob sie an einer Klausur im Fach Datenbanken teilgenommen haben. Wenn ja, mit dem Klausurdatum, wenn nein stattdessen mit NULL oder ähnlich

SELECT DISTINCT Student.Name, Klausur.Klausurdatum

FROM Vorlesung

LEFT JOIN Klausur ON KLausur.Vorlesungsname = 'DB'

LEFT JOIN Schreibt ON Schreibt.Klausurschlüssel = klausur.Klausurschlüssel

FULL JOIN Student ON STUDENT.MATRNR = SCHREIBT.MATRNR;

**1:1** (1 zu 1)

**1:c** (1 zu [0 oder 1])

**1:m** (1 zu [mindestens 1])

**1:mc** (1 zu [beliebig vielen])

**c:c** ([1 oder 0] zu [0 oder 1]; entspricht *1:1* in Chen-Notation)

**c:m** ([0 oder 1] zu [mindestens 1])

**c:mc** ([0 oder 1] zu [beliebig vielen]; entspricht *1:n* in Chen-Notation)

**m:m** ([mindestens 1] zu [mindestens 1])

**m:mc** ([mindestens 1] zu [beliebig vielen])

**mc:mc** ([beliebig viele] zu [beliebig vielen]; entspricht *m:n* in Chen-Notation)

1. Jedes Attribut ist vom gewählten Primärschlüssel funktional abhängig und »atomar«.
2. Sind alle Nicht-Schlüsselatribute vom kompletten Schlüsselatribut abhänig (nicht nur von teilen).
   * R(A, B, C, D) mit {A, B} → C und B → D 🡪 In die 2. Normalform 🡪 R1(A,B,C) R2(B, D)
3. Kein Nicht-Schlüsselatribut ist abhängig von einem Nicht-Schlüsselatribut.
   * R(A, B, C, D) A → B, A → C und C → D sowie A → D, A → C → D 🡪 In die 3. Normalform 🡪 R1(A, B, C) R2(C, D)

Kunden(KundenNr, Name)

Bestellungen(BestellNr, KundenNr, Bestelldatum, Lieferdatum)

Bestellposition(BestellNr, ArtikelNr, Menge)

Artikel(ArtikelNr, Bezeichnung, Preis)

1. Kunden ohne Bestellungen, Name ausgeben:

SELECT Name FROM Kunden WHERE KundenNr NOT IN (SELECT KundenNr FROM Bestellungen);

1. Wieviele Bestellungen pro Kunde, Name und Anzahl ausgeben (ohne Kunden, die Bestellungen haben):

SELECT Name, Count(BestellNr) AS Anzahl FROM Kunden, Bestellungen WHERE Kunden.KundenNr = Bestellungen.KundenNr GROUP BY Name;

1. Bezeichnung aller Artikel, die ‘Donald‘ bestellt hat.

SELECT Bezeichnung FROM Artikel, Bestellposition, Bestellungen, Kunden WHERE Name=‘Donald‘ AND Artikel.ArtikelNr = Bestellposition.ArtikelNr AND Bestellposition.BestellNr = Bestellungen.BestellNr AND Bestellungen.KundenNr = Kunden.KundenNr;

Eine **Person** kann Mitliged in keinem oder mehreren **Vereinen** sein. Ein Verein muss mindestens eine Person als Mitglied haben.

Eine **Person** ist entweder ein oder kein Lieferant. Ein **Lieferant** ist immer zwingend eine Person

|  |  |  |
| --- | --- | --- |
| CREATE TABLE Personen (     PersonID number NOT NULL,     Nachname varchar(255),     Vorname varchar(255),     Age number (2),  CONSTRAINT PK\_Personen = PRIMARY KEY (PersonID) ); | CREATE TABLE Vereine (     Gründung number (4),     Vorsitz varchar(255),     Ort varchar(255) ,     VereinsNr number NOT NULL,  CONSTRAINT PK\_Vereine = PRIMARY KEY (VereinsNr) ); | CREATE TABLE Mitglied (     PersonID number NOT NULL,     VereinsNr number NOT NULL,     Eintritt date,  CONSTRAINT PK\_Mitglied = PRIMARY KEY (PersonID, VereinsNr)  CONSTRAINT FK1\_Mitglied = FOREIGN KEY (PersonID) REFERENCES Personen(PersonenID)  CONSTRAINT FK2\_Mitglied = FOREIGN KEY (VereinsNr) REFERENCES Vereine(VereinsNr) ); |
| CREATE TABLE Lieferanten (     PersonID number NOT NULL, UNIQUE,  LieferantenNr number,  CONSTRAINT PK\_ Lieferanten = PRIMARY KEY (LieferantenNr)  CONSTRAINT FK\_ Lieferanten = FOREIGN KEY (PersonID) REFERENCES Personen(PersonenID) ); | |

Automatisch generierte PersonalID

CREATE VIEW WiMiInfo (Name, PersonalID, Departmentname, Fachgebiet)

AS SELECT Mitarbeiter.Name, Mitarbeiter.PersonalID, Mitarbeiter.Departmentname, Wis\_Mitarbeiter.Fachgebiet

FROM Mitarbeiter, Wis\_Mitarbeiter WHERE Mitarbeiter.PersonalID = Wis\_Mitarbeiter.PersonalID;

CREATE OR REPLACE TRIGGER insert\_WiMiInfo INSTEAD OF INSERT ON WiMiInfo

DECLARE x INTEGER;

BEGIN

SELECT COUNT(\*) + 1 INTO x

FROM Mitarbeiter;

INSERT INTO Mitarbeiter (personalid, departmentname) VALUES(x, :new.departmentname);

INSERT INTO Wis\_Mitarbeiter (personalid, fachgebiet) VALUES(x, :new.fachgebiet);

Überprüfung ob bei einer Besetellung das Lieferdatum nach dem Bestelldatum ist.

CREATE TRIGGER Datumsprüfung BEFORE INSERT ON Bestellungen FOR EACH ROW

BEGIN

IF :NEW.Lieferdatum <= :NEW.Bestelldatum

THEN

RAISE\_APPLICATION\_ERROR(-20001, ‘Lieferdatum falsch ‘);

END IF;

END;