

SQL-Befehle

In Kapitel 25 stehen zehn Beispiele für SQL-Befehle, die sich auf die Tabellen von Kapitel 9 beziehen. Hier finden Sie den Code dazu.

1.

Gesucht sind die Namen der Teilnehmer, deren Geburtsjahr zwischen 2000 und 2011 liegt, die in Berlin oder Potsdam wohnen, deren Nachname mit »Mey« anfängt und die keine Ermäßigung bekommen.

```
SELECT Vorname, Name FROM Teilnehmer
WHERE EXTRACT(YEAR FROM Geburtstag) BETWEEN 2001 AND 2010
AND Ort IN ('Berlin', 'Potsdam')
AND Name LIKE 'Mey%'
AND Ermässigung <> 'j';
```

2.

In welchen Sparten hat der Teilnehmer Volker Viellern im Sommersemester 2024 Kurse belegt? Gefragt ist nach den Spartenbezeichnungen, nicht nach dem Kürzel.

```
SELECT DISTINCT Sparten.Bezeichnung
FROM Sparten, Kurse, Angebote, Semester, Anmeldungen,
Teilnehmer
WHERE Vorname = 'Volker'
AND Name = 'Viellern'
AND Teilnehmer.TeilnehmerNr = Anmeldungen.TeilnehmerNr
AND Anmeldungen.KursNr = Angebote.KursNr
AND Anmeldungen.Semesterkürzel = Angebote.Semesterkürzel
AND Semester.Bezeichnung = 'Sommersemester 2024'
AND Semester.Semesterkürzel = Angebote.Semesterkürzel
AND Angebote.KursNr = Kurse.KursNr
AND Sparte = Kürzel;
```

3.

Gefragt sind die Namen der Teilnehmer und die Bezeichnungen der Kurse, zu denen sie sich angemeldet haben. Personen, die sich noch nie angemeldet haben, sollen auch einmal ausgegeben werden.

```
SELECT Vorname, Name, Kurse.Bezeichnung FROM Kurse
NATURAL JOIN Angebote
NATURAL JOIN Anmeldungen
```

```
RIGHT JOIN Teilnehmer
ON Teilnehmer.TeilnehmerNr = Anmeldungen.TeilnehmerNr;
```

4.

Es sollen die Kurse im Sommersemester 2024 soll ausgegeben werden, für die es weniger als 6 Anmeldungen gibt.

```
SELECT Kurse.Bezeichnung, COUNT(AnmeldeNr)
FROM Kurse, Angebote, Semester, Anmeldungen
WHERE Kurse.KursNr = Angebote.KursNr
AND Angebote.Semesterkürzel = Semester.Semesterkürzel
AND Angebote.KursNr = Anmeldungen.KursNr
AND Angebote.Semesterkürzel = Anmeldungen.Semesterkürzel
AND Semester.Bezeichnung = 'Sommersemester 2024'
GROUP BY Kurse.Bezeichnung HAVING COUNT(AnmeldeNr) < 6;
```

Sie sollten zum Testen dieser Abfrage mindestens einen Kurs haben, der im Sommersemester 2024 mindestens 6 Anmeldungen hat.

5.

Gefragt ist nach den Kursen, die im Sommersemester 2024 am Freitag, Samstag oder Sonntag stattfinden. Ausgegeben werden sollen die Kursbezeichnung, der vollständige Wochentag sowie die Sparte, der der Kurs zugeordnet ist. Wenn es keine zugeordnete Sparte gibt, soll »keiner Sparte zugeordnet« ausgegeben werden.

```
SELECT Kurse.Bezeichnung,
CASE
    WHEN Wochentag = 'Fr' THEN 'Freitag'
    WHEN Wochentag = 'Sa' THEN 'Samstag'
    WHEN Wochentag = 'So' THEN 'Sonntag'
END AS Wochentag,
COALESCE(Sparten.Bezeichnung, 'keiner Sparte zugeordnet') AS
Sparte
FROM Kurse, Angebote, Sparten, Semester
WHERE Sparte = Kürzel
AND Kurse.KursNr = Angebote.KursNr
AND Angebote.Semesterkürzel = Semester.Semesterkürzel
AND Semester.Bezeichnung = 'Sommersemester 2024'
AND Wochentag IN ('Fr', 'Sa', 'So');
```

6.

Geben Sie die Namen der Teilnehmer aus, die sich im Sommersemester zu einem Kurs angemeldet haben. Es soll der 1. Buchstabe des Vornamens, der Name, die Kursbezeichnung und der ganzzahlig gerundete Prozentsatz einer eventuell gewährten Ermäßigung ausgegeben werden.

```
SELECT CONCAT(SUBSTR(Vorname, 1, 1), '. ', Name) AS Name,
Kurse.Bezeichnung,
CASE
    WHEN Ermässigung = 'j' THEN ROUND((Normalpreis - Ermässigt)
    / Normalpreis * 100)
    ELSE 0
END AS Ermässigung
FROM Kurse, Angebote, Anmeldungen, Teilnehmer, Semester
WHERE Kurse.KursNr = Angebote.KursNr
AND Angebote.Semesterkürzel = Semester.Semesterkürzel
AND Semester.Bezeichnung = 'Sommersemester 2024'
AND Angebote.KursNr = Anmeldungen.KursNr
AND Anmeldungen.TeilnehmerNr = Teilnehmer.TeilnehmerNr
AND Angebote.Semesterkürzel = Anmeldungen.Semesterkürzel
AND Anmeldungen.TeilnehmerNr = Teilnehmer.TeilnehmerNr;
```

7.

Wir wollen wissen, wie hoch die gesamten eingenommenen Teilnehmergebühren für das Sommersemester 2024 sind.

```
SELECT
SUM(Ermässigt *
    (SELECT COUNT(AnmeldeNr) FROM Anmeldungen, Teilnehmer
    WHERE Anmeldungen.KursNr = Angebote.KursNr
    AND Anmeldungen.Semesterkürzel = Angebote.Semesterkürzel
    AND Anmeldungen.TeilnehmerNr = Teilnehmer.TeilnehmerNr
    AND Ermässigung = 'j'))
+
SUM(Normalpreis *
    (SELECT COUNT(AnmeldeNr) FROM Anmeldungen, Teilnehmer
    WHERE Anmeldungen.KursNr = Angebote.KursNr
    AND Anmeldungen.Semesterkürzel = Angebote.Semesterkürzel
    AND Anmeldungen.TeilnehmerNr = Teilnehmer.TeilnehmerNr
    AND Ermässigung <> 'j'))
AS Gebühren
FROM Angebote
WHERE Semesterkürzel =
```

```
(SELECT Semesterkürzel FROM Semester
WHERE Bezeichnung = 'Sommersemester 2024');
```

8.

Wir wollen die Teilnehmer wissen, die sich im Sommersemester 2024 zu keiner Veranstaltung angemeldet haben. Oder anders ausgedrückt, für die es in dem Semester keine Anmeldungen gibt.

```
SELECT Vorname, Name FROM Teilnehmer
WHERE NOT EXISTS
(SELECT AnmeldeNr FROM Anmeldungen, Angebote, Semester
WHERE Anmeldungen.TeilnehmerNr = Teilnehmer.TeilnehmerNr
AND Anmeldungen.KursNr = Angebote.KursNr
AND Anmeldungen.Semesterkürzel = Angebote.Semesterkürzel
AND Angebote.Semesterkürzel = Semester.Semesterkürzel
AND Semester.Bezeichnung = 'Sommersemester 2024');
```

9.

Nehmen wir an, dass sehr oft die Anzahl der Anmeldungen für einen Kurs abgefragt wird. Sie kommen also auf die Idee, in der Tabelle Angebote ein Attribut Teilnehmeranzahl einzuführen.

```
ALTER TABLE Angebote ADD COLUMN Teilnehmeranzahl INTEGER;
```

```
UPDATE Angebote SET Teilnehmeranzahl =
(SELECT COUNT(*) FROM Anmeldungen
WHERE KursNr = Angebote.KursNr
AND Semesterkürzel = Angebote.Semesterkürzel);
```

```
CREATE TRIGGER Anmeldungen_Kursteilnehmer
AFTER INSERT OR UPDATE OR DELETE ON Anmeldungen FOR EACH ROW
BEGIN
    IF NEW.AnmeldeNr IS NOT NULL
    THEN
        UPDATE Angebote SET Teilnehmeranzahl = Teilnehmeranzahl + 1
        WHERE KursNr = NEW.KursNr
        AND Semesterkürzel = NEW.Semesterkürzel;
    END IF;
    IF OLD.AnmeldeNr IS NOT NULL
    THEN
        UPDATE Angebote SET Teilnehmeranzahl = Teilnehmeranzahl - 1
        WHERE KursNr = OLD.KursNr
```

```
    AND Semesterkürzel = OLD.Semesterkürzel;  
    END IF;  
END;
```

Dieser Trigger ist in MySQL-Syntax formuliert. Bei anderen DBMS ist er gegebenenfalls anzupassen. Für PostgreSQL sieht er beispielsweise so aus:

```
CREATE OR REPLACE FUNCTION Update_Teilnehmeranzahl()  
    RETURNS TRIGGER  
    LANGUAGE PLPGSQL  
AS $$  
BEGIN  
    IF NEW.AnmeldeNr IS NOT NULL  
    THEN  
        UPDATE Angebote SET Teilnehmeranzahl = Teilnehmeranzahl + 1  
        WHERE KursNr = NEW.KursNr  
        AND Semesterkürzel = NEW.Semesterkürzel;  
    END IF;  
    IF OLD.AnmeldeNr IS NOT NULL  
    THEN  
        UPDATE Angebote SET Teilnehmeranzahl = Teilnehmeranzahl - 1  
        WHERE KursNr = OLD.KursNr  
        AND Semesterkürzel = OLD.Semesterkürzel;  
    END IF;  
    RETURN NEW;  
END; $$
```

```
CREATE TRIGGER Teilnehmer_Trigger  
BEFORE INSERT OR DELETE OR UPDATE ON Anmeldungen  
FOR EACH ROW  
EXECUTE FUNCTION Update_Teilnehmeranzahl();
```

10.

Wir wollen je Semester eine Rangfolge der Kurse nach den Teilnehmerzahlen ausgeben.

```
SELECT Semesterkürzel, KursNr,  
RANK() OVER (PARTITION BY Semesterkürzel ORDER BY  
Teilnehmeranzahl DESC) Rang  
FROM Angebote;
```