

Auswahl aller Spalten

SELECT * FROM spieler

	id	titel	name	vorname	geboren	geschlecht	
1	2	(null)	Elfers	Rainer	09.12.1991 00:00:00	M	Göp
2	6	(null)	Peters	Robert	04.05.1991 00:00:00	M	Göp
3	7	(null)	Wiegand	Günther	26.08.1981 00:00:00	M	Göp
4	8	(null)	Neuhaus	Berta	05.09.1989 00:00:00	W	Uhi
5	27	(null)	Kohl	Dagmar	14.11.1972 00:00:00	W	Rec
6	28	(null)	Kohl	Claudia	01.05.1968 00:00:00	W	Jeb
7	39	(null)	Bischof	Dennis	09.01.1969 00:00:00	M	Göp
8	44	Dr.	Bäcker	Egon	03.04.1990 00:00:00	M	Uhi
9	57	von	Böhmen	Manfred	19.12.1994 00:00:00	M	Göp
10	83	(null)	Hofmann	Philipp	03.04.1983 00:00:00	M	Göp
11	95	(null)	Müller	Paul	09.07.1986 00:00:00	M	Fau
12	100	(null)	Peters	Franz	03.05.1983 00:00:00	M	Göp
13	104	(null)	Maurer	Doris	03.09.1990 00:00:00	W	Rec
14	112	von	Bauer	Irene	19.12.1990 00:00:00	W	Eis

- mit * werden alle Attribute einer Relation ausgewählt.
- Bezieht sich die Abfrage nur auf eine Relation, ist das Ergebnis eine Kopie der Relation selbst.
- Einbezug mehrere Tabellen (JOIN) → später!

Projektion

SELECT name, vorname FROM spieler

	name	vorname
1	Elfers	Rainer
2	Peters	Robert
3	Wiegand	Günther
4	Neuhaus	Berta
5	Kohl	Dagmar
6	Kohl	Claudia
7	Bischof	Dennis
8	Bäcker	Egon
9	Böhmen	Manfred
10	Hofmann	Philipp
11	Müller	Paul
12	Peters	Franz
13	Maurer	Doris
14	Bauer	Irene

- Es können einzelne Attribute angegeben werden, die in die Ergebnisrelation einbezogen werden sollen.
(Projektion)
- Die Ergebnisrelation enthält dann nur die angegebenen Attribute.

Ausdrücke & Konstante

Statt einzelner Attribute können dem SELECT-Statement auch Ausdrücke und Konstante übergeben werden (z.B. für einfache Berechnungen).

Bei den Berechnungen / Umformungen können die Werte der Attribute selbstverständlich einbezogen werden.

SELECT 20 / 3;

column1
1
6



Nachkommestellen werden bei einer Integerdivision abgeschnitten

**SELECT id+kapitaen_nr
FROM team;**

column1
1
28
2
47
3
106

**SELECT spieler+22,
'Abc'+name FROM spieler;**

	column1	column2
1	24	AbcElfers
2	28	AbcPeters
3	29	AbcWiegand
4	30	AbcNeuhaus
5	49	AbcKohl
6	50	AbcKohl
7	61	AbcBischof
8	66	AbcBäcker
9	79	AbcBöhmen
10	105	AbcHofmann
11	117	AbcMüller
12	122	AbcPeters
13	126	AbcMaurer
14	134	AbcBauer

Benennung von Ergebnis-Spalten

```
SELECT name AS Nachname FROM spieler;
```

	Nachname
1	Elfers
2	Peters
3	Wiegand
4	Neuhaus
5	Kohl
6	Kohl
7	Bischof
8	Bäcker
9	Böhmen
10	Hofmann
11	Müller
12	Peters
13	Maurer
14	Bauer

```
SELECT 20 / 3 AS Ergebnis;
```

	Ergebnis
1	6

- Die Spaltennamen der Ergebnisrelation können mit dem Schlüsselwort AS beliebig umbenannt werden.

Rechenoperationen

+ Addition

- Subtraktion

* Multiplikation

/ Division

% Modulo

() Klammersetzung

Punkt-vor-Strich Regel

Datums- & Zeitfunktionen

day()	Ermittelt den Tag eines Datumswerts
month()	Ermittelt den Monat eines Datumswerts
year()	Ermittelt das Jahr eines Datumswerts
getdate()	Liefert aktuelle Zeit und aktuelles Datum
datediff()	Berechnet die Zeitliche Differenz zwischen zwei Zeitpunkten

Datums- & Zeitfunktionen

```
SELECT name, vorname,
       datediff(yy, geboren, getdate()) AS 'Alter'
FROM spieler;
```

	name	vorname	Alter
1	Elfers	Rainer	19
2	Peters	Robert	19
3	Wiegand	Günther	29
4	Neuhaus	Berta	21
5	Kohl	Dagmar	38
6	Kohl	Claudia	42
7	Bischof	Dennis	41
8	Bäcker	Egon	20
9	Böhmen	Manfred	16
10	Hofmann	Philipp	27
11	Müller	Paul	24
12	Peters	Franz	27
13	Maurer	Doris	20
14	Bauer	Irene	20

Notation

Zeichenketten (Strings):

- MSSQL: Einfache Anführungszeichen
 - 'Göppingen'

Datumswerte:

- MSSQL:
 - 'dd/mm/yy'
 - 'dd.mm.yyyy'
- MySQL:
 - 'yyyy-mm-dd'

Dezimalzahlen werden mit einem Punkt statt einem Komma geschrieben.

Vergleichsoperatoren

```

SELECT name FROM spieler WHERE ort = 'Göppingen';
SELECT name FROM spieler WHERE year(geboren) > 1990;
SELECT name FROM spieler WHERE id < 15;
SELECT name FROM spieler WHERE plz <= 73035;
SELECT name FROM spieler WHERE hausnummer >= 10;
SELECT name FROM spieler WHERE name <> 'Kohl';
SELECT name FROM spieler
    WHERE hausnummer BETWEEN 5 AND 20;
SELECT name FROM spieler
    WHERE hausnummer IN (4, 8 , 16);
SELECT name FROM spieler WHERE name LIKE 'B%';
SELECT name FROM spieler WHERE name LIKE '_au%';
SELECT name FROM spieler WHERE titel IS NULL;
    
```

Vergleichsoperatoren

Operatoren Erklärung

= Attributwert gleicht einem anderen Attributwert oder einer Konstanten

< > <= >= Attribut soll kleiner, größer, kleiner gleich oder größer gleich einem anderen Attributwert oder einer Konstanten sein

<> Attributwert ist ungleich einem anderen Attributwert oder einer Konstanten

BETWEEN Attributwert zwischen zwei Grenzen

IN Attributwert in einer Menge enthalten

LIKE Suche nach Zeichenketten anhand von Ähnlichkeitsoperatoren:
% Platzhalter für eine beliebige Zeichenkette
_ Platzhalter für ein Zeichen

IS NULL **IS NULL** oder **IS NOT NULL** zur Selektion nicht definierter Attributwerte

Vergleichsoperatoren

```
SELECT spielernr, name, vorname, strasse, ort
FROM spieler
WHERE
```

```
    strasse NOT LIKE '%gärten' AND
    ort='Göppingen' OR spielernr=28
```

	id	name	vorname	strasse	ort
1	2	Elfers	Rainer	Blumenstraße	Göppingen
2	6	Peters	Robert	Ziegelstraße	Göppingen
3	7	Wiegand	Günther	Panoramastraße	Göppingen
4	28	Kohl	Claudia	Auf dem Hof	Jebenhausen
5	39	Bischof	Dennis	Hohensteinstraße	Göppingen
6	57	Böhmen	Manfred	Seefridstraße	Göppingen
7	83	Hofmann	Philipp	Heilbronner Straße	Göppingen

- Logische Aussagen können mit AND oder OR logisch miteinander verknüpft werden.
 - Klammersetzung wird dabei beachtet.
- Mit NOT wird der Wahrheitsgehalt einer logischen Aussage umgedreht

ORDER BY

```
SELECT name, vorname FROM spieler
ORDER BY name DESC, vorname ASC, ort;
```

	name	vorname
1	Wiegand	Günther
2	Peters	Franz
3	Peters	Robert
4	Neuhaus	Berta
5	Müller	Paul
6	Maurer	Doris
7	Kohl	Claudia
8	Kohl	Dagmar
9	Hofmann	Philipp
10	Elfers	Rainer
11	Böhmen	Manfred
12	Bischof	Dennis
13	Bauer	Irene
14	Bäcker	Egon

- Die Ergebnisrelation kann mit dem Schlüsselwort ORDER BY nach Attributwerten mehrstufig sortiert werden.
 - ASC aufsteigende Sortierung (vorgabe)
 - DESC absteigende Sortierung
- Es kann auch nach nicht gewählten (aber implizit vorhandenen) Attributen sortiert werden.

SELECT

Grundlegende Syntax: *(vereinfacht)*

```

SELECT      [ ALL | DISTINCT ]
            * | Spaltenname [ [ AS ] alias ], ...
            | ausdruck

FROM        tabellenname [ [ AS ] alias ], ...

[ WHERE     bedingung ]

[ GROUP BY  spaltenname, ... ]

[ HAVING    bedingung ]

[ ORDER BY  spaltenname [ ASC | DESC ], ... ]
    
```

SELECT

SELECT

← Was will ich sehen

FROM

← Wo kommen die Daten her

WHERE

← welche Bedingungen müssen
erfüllt sein

ORDER BY

← wie sollen die Daten sortiert werden

1. Einführung

2. Datenbankentwurf

3. Datenbankimplementierung

4. Physische Datenorganisation

5. Anfrageoptimierung

6. Transaktionsverwaltung

7. Datensicherheit und Wiederherstellung

8. Business Intelligence

Übungsaufgabe 1

Welche Spieler wohnen nicht in Göppingen?

Übungsaufgabe 1

Welche Spieler wohnen nicht in Göppingen?

	name	vorname
1	Neuhaus	Berta
2	Kohl	Dagmar
3	Kohl	Claudia
4	Bäcker	Egon
5	Müller	Paul
6	Maurer	Doris
7	Bauer	Irene

Übungsaufgabe 2

Wie heißen die Spieler die in Jebenhausen oder Uhingen wohnen?

```
1 SELECT name, vorname FROM dbo.Boehmisch_spieler
2 WHERE ort='Jebenhausen' OR ort='Uhingen';
```

Übungsaufgabe 2

Wie heißen die Spieler die in Jebenhausen oder UHINGEN wohnen?

	name	vorname
1	Neuhaus	Berta
2	Kohl	Claudia
3	Bäcker	Egon

1. Einführung

2. Datenbankentwurf

3. Datenbankimplementierung

4. Physische Datenorganisation

5. Anfrageoptimierung

6. Transaktionsverwaltung

7. Datensicherheit und Wiederherstellung

8. Business Intelligence

Übungsaufgabe 3

Welche Spieler sind 1995, 2000 oder 2006 beigetreten?

Übungsaufgabe 3

Welche Spieler sind 1995, 2000 oder 2006 beigetreten?

	name	vorname
1	Wiegand	Günther
2	Neuhaus	Berta
3	Bäcker	Egon
4	Böhmen	Manfred
5	Hofmann	Philipp
6	Bauer	Irene

Übungsaufgabe 4

Welche Spieler haben in der zweiten Jahreshälfte Geburtstag? Die Antworttabelle soll nach dem Monat gruppiert sein.

```
1 SELECT name, vorname, month(geboren) AS 'Geburtsmonat' FROM dbo.Boehmisch_spieler
2 WHERE month(geboren) BETWEEN 6 AND 12
3 ORDER BY 'Geburtsmonat' ASC;
```

Übungsaufgabe 4

Welche Spieler haben in der zweiten Jahreshälfte Geburtstag? Die Antworttabelle soll nach dem Monat gruppiert sein.

Ergebnisse		Meldungen	
	name	vorname	Geburtsmonat
1	Müller	Paul	7
2	Wiegand	Günther	8
3	Neuhaus	Berta	9
4	Maurer	Doris	9
5	Kohl	Dagmar	11
6	Böhmen	Manfred	12
7	Bauer	Irene	12
8	Elfers	Rainer	12

Übungsaufgabe 5

Welche Spielernummern haben die Spieler, deren Nachnamen mit M anfangen?

```
1 SELECT spielernr FROM dbo.Boehmisch_spieler
2 WHERE name LIKE 'M%';
```

Übungsaufgabe 5

Welche Spielernummern haben die Spieler, deren Nachnamen mit M anfangen?

	spielernr
1	95
2	104

Spalten- & Aggregatfunktionen

Spalten- Aggregatfunktionen werden auf ganzen Spalten bzw. Wertegruppierungen angewandt und liefern jeweils nur **exakt einen Wert** zurück.

Beispiel für ein Fehlerhaftes SQL Statement:

```
SELECT name, count(*)  
FROM spieler
```

Viele Werte
ein Wert



„Passt nicht zusammen!“
(Besser: „Würde 1NF verletzt“)

```
SELECT count(*) FROM  
spieler
```

column1
1
14

```
SELECT max(hausnummer)  
FROM spieler  
WHERE ort='Göppingen';
```

```
SELECT sum(strafe)  
FROM strafe  
WHERE year(datum)<2002;
```

Spalten- & Aggregatfunktionen

Funktion Erklärung

avg()	Mittelwert einer Spalte / Gruppierung
count()	Anzahl aller Tupel einer Ergebnisrelation / Gruppierung (Tupel die NULL enthalten werden nicht gezählt)
max()	Größter Wert der Spalte / Gruppierung
min()	Kleinsten Wert der Spalte / Gruppierung
sum()	Summe der Werte einer Spalte / Gruppierung

Übungsaufgabe 6

Wie hoch war die durchschnittliche Strafe 2003?

```
SELECT AVG(strafe) FROM dbo.Boehmisch_strafe  
WHERE year(datum)=2003;
```

Übungsaufgabe 6

Wie hoch war die durchschnittliche Strafe 2003?

	(Kein Spaltenname)
1	22,50

Übungsaufgabe 7

Wie viele Spieler sind erst nach dem 20. Lebensjahr dem Verein beigetreten?

```
SELECT count(spielernr) FROM dbo.Boehmisch_spieler  
WHERE datediff(yy,geboren,beitritt)>20;
```

Übungsaufgabe 7

Wie viele Spieler sind erst nach dem 20. Lebensjahr dem Verein beigetreten?

	(Kein Spaltenname)
1	1

Übungsaufgabe 8

Wie viele Spieler sind zwischen 20 und 30 Jahre alt?

```
SELECT count(spielernr) AS 'Spieler zw. 20 und 30' FROM dbo.Boehmisch_spieler  
WHERE datediff(yy, geboren, beitrtritt) BETWEEN 20 AND 30;
```

Übungsaufgabe 8

Wie viele Spieler sind zwischen 20 und 30 Jahre alt?

Ergebnisse		Meldungen	
		Spieler zw. 20 und 30	
1		1	

GROUP BY

Mit **GROUP BY** werden diejenigen Tupel zu Gruppen zusammengefasst, die bezüglich aller Attribute, die hinter der Klausel aufgelistet sind, die gleichen Werte aufweisen.

Es lassen sich Aggregatfunktionen in die Auswahl aufnehmen, die sich sodann auf die Werte **jeder einzelnen Gruppierung** beziehen

**SELECT ort
FROM spieler
ORDER BY ort**

	ort
1	Eislingen/Fils
2	Faurndau
3	Göppingen
4	Jebenhausen
5	Rechberghausen
6	Uhingen

	ort
1	Eislingen/Fils
2	Faurndau
3	Göppingen
4	Göppingen
5	Göppingen
6	Göppingen
7	Göppingen
8	Göppingen
9	Göppingen
10	Jebenhausen
11	Rechberghausen
12	Rechberghausen
13	Uhingen
14	Uhingen

**SELECT ort FROM spieler
GROUP BY ort**

	ort	column2
1	Eislingen/Fils	1
2	Faurndau	1
3	Göppingen	7
4	Jebenhausen	1
5	Rechberghausen	2
6	Uhingen	2

**SELECT ort, count(*) FROM
spieler GROUP BY ort**

SELECT

SELECT

← Was will ich sehen

FROM

← Wo kommen die Daten her

WHERE

← welche Bedingungen müssen erfüllt sein

GROUP BY

← wonach soll Gruppirt werden

ORDER BY

← wie sollen die Daten sortiert werden

Übungsaufgabe 9

Wie viele Männer/Frauen/Div. hat der Verein?

```
SELECT geschlecht, count(spielernr) FROM dbo.Boehmisch_spieler  
GROUP BY geschlecht;
```

Übungsaufgabe 9

Wie viele Männer/Frauen/Div. hat der Verein?

	geschlecht	(Kein Spaltenname)
1	M	9
2	W	5

Übungsaufgabe 10

Wie alt sind durchschnittlich die Vereinsmitglieder?

Unterscheiden sie nach Geschlecht!

```
1 -- Das aktuelle Alter unterscheidet sich von der Lösung in den Folien
2 -- da diese zu einem früheren Zeitpunkt erstellt wurden
3 -- trotzdem ist diese Lösung nicht die genaueste, da nur die Differenz der
4 -- Jahre berücksichtigt aber nicht ob die Person zum aktuellen Zeitpunkt schon
5 -- Geburtstag hatte
6
7 -- SELECT geschlecht,
8 --         AVG(DATEDIFF(yy, geboren, GETDATE())) AS [Durchschn. Alter]
9 -- FROM dbo.Boehmisch_spieler
10 -- GROUP BY geschlecht;
```

Übungsaufgabe 10

Wie alt sind durchschnittlich die Vereinsmitglieder?

Unterscheiden sie nach Geschlecht!

<div> <div>Ergebnisse</div> <div>Meldungen</div> </div>		
	geschlecht	Durchschn. Alter
1	M	37
2	W	41

Übungsaufgabe 11

Erstellen sie eine Liste von Spielern, die eine Strafe zahlen mussten, wobei die Liste absteigend nach dem Gesamtbetrag sortiert sein soll!
(Hinweis: Die Spielernummer reicht als Ergebnis aus!)

```
1 -- die Werte in der Beispielausgabe in den Übungsgaben stimmen nicht überein
2 -- externes nachzählen der Werte zeigt aber dass die Abfrage korrekt ist
3 SELECT spielernr,
4        SUM(strafe) AS gesamtstrafe
5 FROM dbo.Boehmisch_strafe
6 GROUP BY spielernr
7 HAVING SUM(strafe) > 0
8 ORDER BY gesamtstrafe DESC;
```

Übungsaufgabe 11

Erstellen sie eine Liste von Spielern, die eine Strafe zahlen mussten, wobei die Liste absteigend nach dem Gesamtbetrag sortiert sein soll!
(Hinweis: Die Spielernummer reicht als Ergebnis aus!)

Ergebnisse Meldungen		
	spielernr	(Kein Spaltenname)
1	57	426.00
2	2	245.00
3	6	188.50
4	104	174.00
5	8	173.00
6	39	157.50
7	100	152.50
8	95	147.00
9	7	144.00
10	28	114.50
11	112	109.50
12	27	107.50
13	44	70.00
14	83	68.00

HAVING

Mit HAVING können Bedingungen für Gruppierungen definiert werden. Nur Gruppierungen, welche die Bedingung erfüllen werden in die Ergebnisrelation aufgenommen.

HAVING verhält sich zu **GROUP BY** wie **WHERE** zu **SELECT**

```
SELECT name, sum(spielernr)
FROM spieler
GROUP BY name;
```

	name	(Kein Spaltenname)
1	Bäcker	44
2	Bauer	112
3	Bischof	39
4	Böhmen	57
5	Elfers	2
6	Hofmann	83
7	Kohl	55
8	Maurer	104
9	Müller	95
10	Neuhaus	8
11	Peters	106
12	Wiegand	7

	name	(Kein Spaltenname)
1	Bauer	112
2	Hofmann	83
3	Maurer	104
4	Müller	95
5	Peters	106

```
SELECT name, sum(spielernr)
FROM spieler
GROUP BY name
HAVING sum(spielernr)>80
```

SELECT

SELECT

← Was will ich sehen

FROM

← Wo kommen die Daten her

WHERE

← welche Bedingungen müssen erfüllt sein

GROUP BY

← wonach soll Gruppirt werden

HAVING

← welche Bedingungen für die Gruppen müssen erfüllt sein

ORDER BY

← wie sollen die Daten sortiert werden

Übungsaufgabe 12

In welchen Städten wohnen mindestens 2 Spieler?

```
1 SELECT ort
2 FROM dbo.Boehmisch_spieler
3 GROUP BY ort
4 HAVING COUNT(spielernr) >=2;
```

Übungsaufgabe 12

In welchen Städten wohnen mindestens 2 Spieler?

	ort
1	Göppingen
2	Rechberghausen
3	Uhingen

(Fiktive) Verarbeitungsreihenfolge

1. FROM: Auswahl der Tabelle
2. WHERE: Selektiert die Tupel, die der Bedingung genügen
3. GROUP BY: Gruppiert die Tupel auf Basis der gleicher Werte
4. HAVING: Selektiert Gruppen die der Bedingung genügen
5. SELECT: Selektiert Attribute
6. ORDER BY: Sortiert die Tupel

SELECT spielernr, sum(strafe) AS "sum(strafe)" FROM strafe WHERE year(datum)>2005 GROUP BY spielernr HAVING count(strafe)>=2 ORDER BY sum(strafe)

	id	spielernr	datum	grund_id	bezahlt_am	strafe
1	1		2.20.12.2004 00:00:00		1.07.03.2008 00:00:00	21,5
2	2		83.11.05.2008 00:00:00		3 (null)	6,5
3	3		57.06.01.2009 00:00:00		3.06.09.2009 00:00:00	43
4	4		100.15.05.2004 00:00:00		3.01.09.2006 00:00:00	13
5	5		39.15.07.2000 00:00:00		4.17.02.2003 00:00:00	32,5
6	6		7.13.07.2004 00:00:00		3 (null)	34,5
7	7		8.04.05.2008 00:00:00		3 (null)	17,5
8	8		2.09.12.2005 00:00:00		3.26.01.2007 00:00:00	6,5
9	9		57.01.09.2008 00:00:00		4 (null)	23,5
10	10		57.09.06.2007 00:00:00		2.21.01.2009 00:00:00	37,5
11	11		95.24.08.2005 00:00:00		3.31.08.2006 00:00:00	30,5
12	12		6.13.03.2007 00:00:00		1.14.04.2008 00:00:00	38
13	13		95.02.11.2005 00:00:00		2.27.08.2009 00:00:00	37
14	14		2.14.10.2008 00:00:00		4 (null)	24

1. FROM

	id	spielernr	datum	grund_id	bezahlt_am	strafe
1	2		83.11.05.2008 00:00:00		3 (null)	6,5
2	3		57.06.01.2009 00:00:00		3.06.09.2009 00:00:00	43
3	7		8.04.05.2008 00:00:00		3 (null)	17,5
4	9		57.01.09.2008 00:00:00		4 (null)	23,5
5	10		57.09.06.2007 00:00:00		2.21.01.2009 00:00:00	37,5
6	12		6.13.03.2007 00:00:00		1.14.04.2008 00:00:00	38
7	14		2.14.10.2008 00:00:00		4 (null)	24
8	15		6.23.06.2009 00:00:00		3 (null)	16
9	16		104.02.12.2007 00:00:00		1 (null)	10,5
10	17		8.09.05.2007 00:00:00		2.07.01.2009 00:00:00	13
11	18		57.27.10.2009 00:00:00		4 (null)	49,5
12	19		2.02.08.2009 00:00:00		2 (null)	32,5
13	20		8.09.07.2006 00:00:00		4.32.08.2009 00:00:00	44,5

2. WHERE

	spielernr	datum	strafe
1		2.25.03.2007 ...	7.50 ...
2		6.05.07.2006 ...	16.00 ...
3		7.27.05.2008 ...	18.50 ...
4		8.09.07.2006 ...	13.00 ...
5		27.13.05.2006 ...	6.00 ...
6		28.29.08.2008 ...	7.00 ...
7		39.13.02.2006 ...	14.50 ...
8		44.08.10.2006 ...	17.50 ...
9		57.09.06.2007 ...	15.50 ...
10		83.11.05.2008 ...	6.50 ...
11		95.30.12.2007 ...	10.00 ...
12		100.12.05.2007 ...	8.00 ...
13		104.14.07.2006 ...	4.00 ...
14		112.18.10.2007 ...	30.50 ...

3. GROUP BY

	spielernr	datum	strafe
1		2.25.03.2007 ...	7.50 ...
2		6.05.07.2006 ...	16.00 ...
3		8.09.07.2006 ...	13.00 ...
4		39.13.02.2006 ...	14.50 ...
5		57.09.06.2007 ...	15.50 ...
6		100.12.05.2007 ...	8.00 ...
7		104.14.07.2006 ...	4.00 ...
8		112.18.10.2007 ...	30.50 ...

5. SELECT

	spielernr	sum(strafe)
1	2	168,5
2	6	138,5
3	8	173
4	39	56,5
5	57	386
6	100	110,5
7	104	208
8	112	109,5

	spielernr	sum(strafe)
1	39	56,5
2	112	109,5
3	100	110,5
4	6	138,5
5	2	168,5
6	8	173
7	104	208
8	57	386

4. HAVING

6. ORDER BY

SQL Syntax (Teil1)

SELECT <Spalten> | * **FROM** <Tabelle>

- Auswahl einiger / aller Spalten aus einer Tabelle

SELECT **DISTINCT** | **ALL**

- Eliminierung doppelter Tupel

SELECT a-5+year('01/19/2005')

- Rechnen mit Ausdrücken / Funktionen

SELECT a **AS** b

- Umbenennen von Ergebnisspalten

ORDER BY x **ASC** | **DESC**

- Ergebnisrelation auf- / absteigend sortieren

WHERE <Bedingung> **AND** | **OR** [**NOT**] <Bedingung>

- Auswahlbedingung, die das Tupel erfüllen soll

GROUP BY a, b

- Gruppierungen über gleiche Attributwerte/kombinationen
- Einsatz von **Aggregat-/Spaltenfunktionen**

HAVING <Bedingung>

- Auswahlbedingung, die die Gruppierung erfüllen soll

Operator

=

> >= <= <

<>

IN

BETWEEN

LIKE

IS NULL

Funktion

avg()

count()

max()

min()

sum()

Zusammenführen von Tabellen (Join)

CROSS Join (Kartesisches Produkt)

- Vollständige Kombination aller Tupel der beteiligten Tabellen.
- Manchmal auch als „einfacher Join“ bezeichnet.

INNER Join (Innerer Verbund)

- Es werden nur Tupel miteinander kombiniert, die über ein entsprechendes Pendant in der anderen Tabelle verfügen.
- Insbesondere
 - Equi-Join (Verbund über paarweise Gleichheit der Werte)
 - Natural Join (natürlicher Verbund)
 - Spezialfall eines Equi-Joins, bei dem nach dem Verbund über Gleichheit doppelte Spalten und Tupel entfernt werden

OUTER Join (Äußerer Verbund)

- Es werden auf jeden Fall alle Tupel der einen Tabelle einbezogen, auch wenn sich kein Pendant in der anderen Tabelle befindet.
- SQL kennt:
 - Left (outer) Join (linksseitiger äußerer Verbund)
 - Right (outer) Join (rechtsseitiger äußerer Verbund)

CROSS Join

SELECT * FROM personen, kinder;

Tabelle: personen

	name	stadt	geboren
1	Andreas	Göppingen	13.12.1987 00:00:00
2	Beate	Uhringen	23.05.1985 00:00:00
3	Claudia	Esslingen	03.12.2003 00:00:00
4	Dietmar	Göppingen	27.02.1956 00:00:00
5	Emil	Göppingen	08.09.1978 00:00:00
6	Franz	Jebenhausen	14.11.1982 00:00:00
7	Gabi	Plochingen	08.03.1967 00:00:00
8	Iris	Göppingen	21.01.2003 00:00:00

Tabelle: kinder

	name	kind
1	Andreas	Franz
2	Andreas	Veronika
3	Dietmar	Iris

Cross Join (Kartesisches Produkt)

	name	stadt	geboren	name	kind
1	Andreas	Göppingen	13.12.1987 00:00:00	Andreas	Franz
2	Beate	Uhringen	23.05.1985 00:00:00	Andreas	Franz
3	Claudia	Esslingen	03.12.2003 00:00:00	Andreas	Franz
4	Dietmar	Göppingen	27.02.1956 00:00:00	Andreas	Franz
5	Emil	Göppingen	08.09.1978 00:00:00	Andreas	Franz
6	Franz	Jebenhausen	14.11.1982 00:00:00	Andreas	Franz
7	Gabi	Plochingen	08.03.1967 00:00:00	Andreas	Franz
8	Iris	Göppingen	21.01.2003 00:00:00	Andreas	Franz
9	Andreas	Göppingen	13.12.1987 00:00:00	Andreas	Veronika
10	Beate	Uhringen	23.05.1985 00:00:00	Andreas	Veronika
11	Claudia	Esslingen	03.12.2003 00:00:00	Andreas	Veronika
12	Dietmar	Göppingen	27.02.1956 00:00:00	Andreas	Veronika
13	Emil	Göppingen	08.09.1978 00:00:00	Andreas	Veronika
14	Franz	Jebenhausen	14.11.1982 00:00:00	Andreas	Veronika
15	Gabi	Plochingen	08.03.1967 00:00:00	Andreas	Veronika
16	Iris	Göppingen	21.01.2003 00:00:00	Andreas	Veronika
17	Andreas	Göppingen	13.12.1987 00:00:00	Dietmar	Iris
18	Beate	Uhringen	23.05.1985 00:00:00	Dietmar	Iris

- Werden in der FROM-Klausel mehrere Tabellen angegeben, so wird grundsätzlich das kartesische Produkt generiert, bei dem alle Elemente vollständig miteinander kombiniert werden → **Cross Join**
- **ACHTUNG!** Spaltennamen sind dann ggf. nicht mehr eindeutig! (vgl. Name!)

Inner Join (hier: Equi-Join)

```
SELECT * FROM personen p, kinder k
WHERE p.name=k.name;
```

Tabelle: personen

	name	stadt	geboren
1	Andreas	Göppingen	13.12.1987 00:00:00
2	Beate	Utingen	23.05.1985 00:00:00
3	Claudia	Esslingen	03.12.2003 00:00:00
4	Dietmar	Göppingen	27.02.1956 00:00:00
5	Emil	Göppingen	08.09.1978 00:00:00
6	Franz	Jebenhausen	14.11.1982 00:00:00
7	Gabi	Plochingen	08.03.1967 00:00:00
8	Iris	Göppingen	21.01.2003 00:00:00

Tabelle: kinder

	name	kind
1	Andreas	Franz
2	Andreas	Veronika
3	Dietmar	Iris

Equi Join

	name	stadt	geboren	name	kind
1	Andreas	Göppingen	13.12.1987 00:00:00	Andreas	Franz
2	Andreas	Göppingen	13.12.1987 00:00:00	Andreas	Veronika
3	Dietmar	Göppingen	27.02.1956 00:00:00	Dietmar	Iris

- Abbildungen eines Inner Joins durch Hinzunahme einschränkender WHERE-Bedingungen (auf die Tupel des Cross Joins)
- Alternative Formulierung *(100% gleichwertig!)*
 - **SELECT ***
FROM personen p INNER JOIN kinder k ON p.name=k.name;

Outer Join

Left (Outer) Join

```
SELECT p.ort, p.name, k.name FROM personen
p LEFT JOIN kinder k ON p.name=k.name
```

Tabelle: personen

	name	stadt	geboren
1	Andreas	Göppingen	13.12.1987 00:00:00
2	Beate	Utingen	23.05.1985 00:00:00
3	Claudia	Esslingen	03.12.2003 00:00:00
4	Dietmar	Göppingen	27.02.1956 00:00:00
5	Emil	Göppingen	08.09.1978 00:00:00
6	Franz	Jebenhausen	14.11.1982 00:00:00
7	Gabi	Plochingen	08.03.1967 00:00:00
8	Iris	Göppingen	21.01.2003 00:00:00

Tabelle: kinder

	name	kind
1	Andreas	Franz
2	Andreas	Veronika
3	Dietmar	Iris

Left Join

	stadt	name	name
1	Göppingen	Andreas	Andreas
2	Göppingen	Andreas	Andreas
3	Utingen	Beate	(null)
4	Esslingen	Claudia	(null)
5	Göppingen	Dietmar	Dietmar
6	Göppingen	Emil	(null)
7	Jebenhausen	Franz	(null)
8	Plochingen	Gabi	(null)
9	Göppingen	Iris	(null)

➤ **Right** Join funktioniert entsprechend!

Join über mehr als zwei Tabellen

SELECT * FROM A, B, C

A		
	A	Wert-A
1	A1	Anton
2	A2	Andreas
3	A3	Anja

B		
	B	WertB
1	B1	Bärbel
2	B2	Bernd

C		
	C	WertC
1	C1	Cäsar
2	C2	Charles
3	C3	Celine

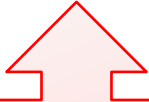
Cross Join (Kartesisches Produkt)

	A	Wert-A	B	WertB	C	WertC
1	A1	Anton	B1	Bärbel	C1	Cäsar
2	A1	Anton	B2	Bernd	C1	Cäsar
3	A1	Anton	B1	Bärbel	C2	Charles
4	A1	Anton	B2	Bernd	C2	Charles
5	A1	Anton	B1	Bärbel	C3	Celine
6	A1	Anton	B2	Bernd	C3	Celine
7	A2	Andreas	B1	Bärbel	C1	Cäsar
8	A2	Andreas	B2	Bernd	C1	Cäsar
9	A2	Andreas	B1	Bärbel	C2	Charles
10	A2	Andreas	B2	Bernd	C2	Charles
11	A2	Andreas	B1	Bärbel	C3	Celine
12	A2	Andreas	B2	Bernd	C3	Celine
13	A3	Anja	B1	Bärbel	C1	Cäsar
14	A3	Anja	B2	Bernd	C1	Cäsar
15	A3	Anja	B1	Bärbel	C2	Charles
16	A3	Anja	B2	Bernd	C2	Charles
17	A3	Anja	B1	Bärbel	C3	Celine
18	A3	Anja	B2	Bernd	C3	Celine

Eindeutigkeit von Spaltennamen

Sind mehrere Tabellen von einer Abfrage betroffen,
sind Spaltennamen ggf. nicht mehr eindeutig:

```
SELECT spielernr FROM spieler, strafe;
```



Diese Abfrage generiert einen **Fehler**,
da **spielernr** in **spieler** und in
strafe vorkommt und daher
mehrdeutig ist!

Spalten mit mehrdeutigen Bezeichnungen daher
immer explizit über den zugehörigen Tabellen-
oder einen Alias-Namen ansprechen:

```
➤ SELECT spieler.spielernr FROM spieler, strafe;
```

oder

```
➤ SELECT sp.spielernr FROM spieler sp, strafe;
```

Join-Übersicht

Syntax:

Cross Join: `SELECT __ FROM a, b`

Inner Join: `SELECT __ FROM a, b WHERE __`

`SELECT __ FROM a INNER JOIN b ON __`

Left Join: `SELECT __ FROM a LEFT JOIN b ON __`

Right Join: `SELECT __ FROM a RIGHT JOIN b ON __`

Abschließende Bemerkungen:

- Relationenmodell kennt grundsätzlich weitere Arten von JOINS → Abbildungen in SQL jedoch grundsätzlich mit den oben genannten Operationen.
- Gleichheitszeichen ist häufigster Join-Operator
→ jeder andere Vergleichsoperator ist aber ebenso möglich:

`SELECT * FROM spieler sp
LEFT JOIN team ON sp.spielernr < 5;`

Wirkungsweise
möge jede(r) selbst
herausfinden!

Join-Übersicht

Nicht vergessen:

Selbstverständlich lassen sich alle vorher vorgestellten Konzepte auch auf die resultierende Relation aus Join-Operationen anwenden!

➤ Also:

Spalten mit SELECT auswählen, mit WHERE weitere Bedingungen festlegen, die Tupel mit GROUP BY gruppieren, Aggregatfunktionen anwenden, Gruppierungen mit HAVING auswählen, das Ergebnis mit ORDER BY sortieren und so weiter und so fort ...

Join-Attribute

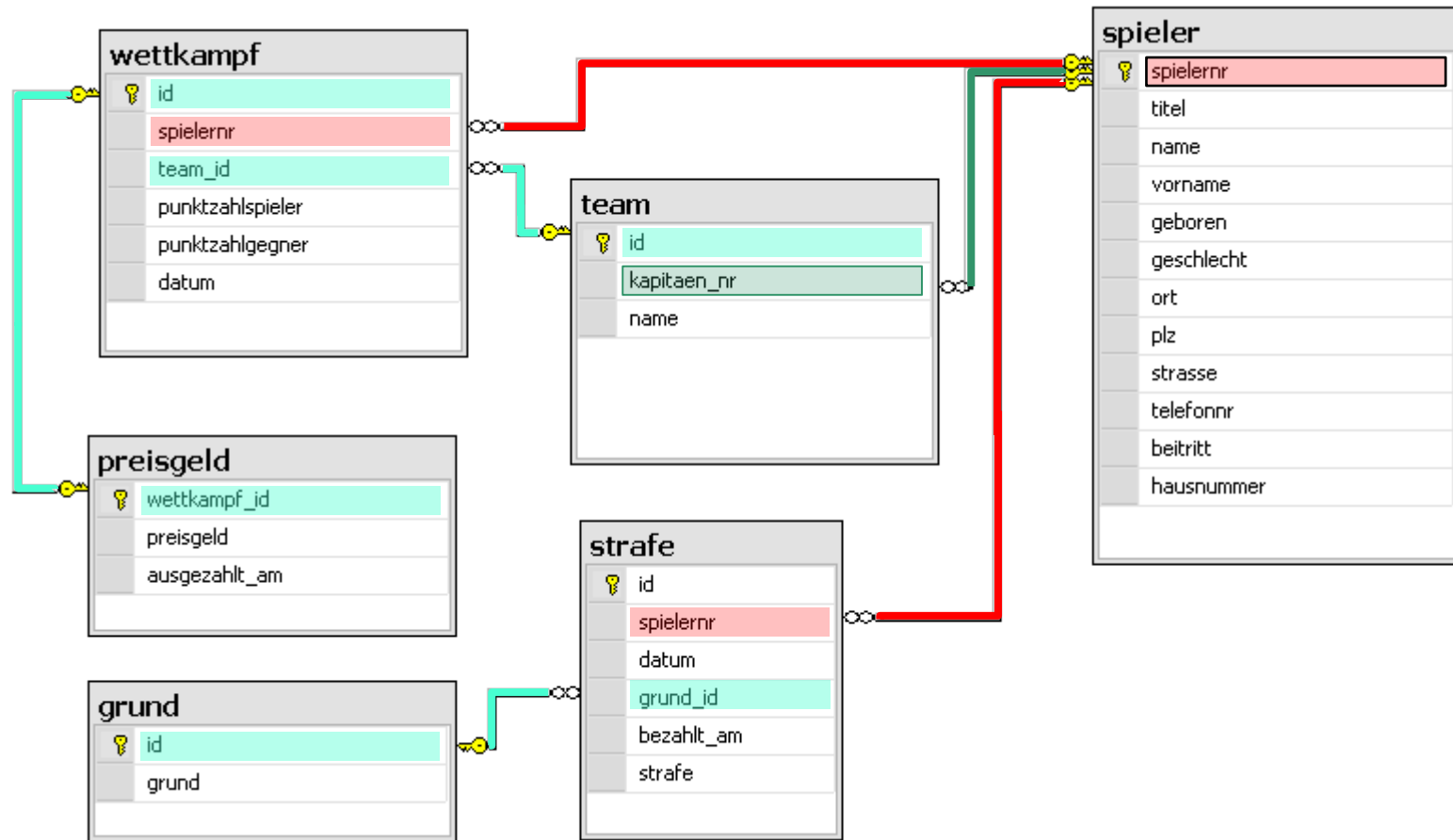
Es gibt verschiedene Arten für die Benennung von Attributen über die gejoint wird.

➤ Gleicher Attributname
z.B. spielernr in Spieler und Wettkampf

➤ <tabellenname>_<PKAttribut>
z.B. wettkampf.team_id bezieht sich auf team.id

➤ ungleiche Attributnamen
(mit etwas Glück sinnzusammenhängend)
z.B. team.kapitaen_nr bezieht sich auf spieler.spielernr

JOIN-Attribute



Übungsaufgabe 13

Welche Spieler haben mindestens eine Strafe erhalten?

```
1 SELECT name, vorname
2 FROM dbo.Boehmisch_spieler AS sp
3 INNER JOIN dbo.Boehmisch_strafe AS st
4 ON sp.spielernr = st.spielernr
5 GROUP BY name, vorname
6 Having count(strafe)>=1;
```

Übungsaufgabe 13

Welche Spieler haben mindestens eine Strafe erhalten?

	name	vorname
1	Neuhaus	Berta
2	Kohl	Claudia
3	Kohl	Dagmar
4	Bischof	Dennis
5	Maurer	Doris
6	Bäcker	Egon
7	Peters	Franz
8	Wiegand	Günther
9	Bauer	Irene
10	Böhmen	Manfred
11	Müller	Paul
12	Hofmann	Philipp
13	Elfers	Rainer
14	Peters	Robert

Übungsaufgabe 14

Wie viele Spiele wurden von den Bären absolviert?

```
1 SELECT count(wt.id)
2 FROM dbo.Boehmisch_wettkampf AS wt
3 INNER JOIN dbo.Boehmisch_team AS tt
4 ON wt.team_id=tt.id
5 WHERE name='Bären';
6
```

Übungsaufgabe 14

Wie viele Spiele wurden von den Bären absolviert?

	(Kein Spaltenname)
1	123

Übungsaufgabe 15

Wie hoch ist die Gesamtstrafe für die Kapitäne der jeweiligen Teams?

```
1 SELECT tt.name AS 'team', sp.name AS 'Spieler_Name', sp.vorname AS 'Spieler_Vorname', sum(st.strafe)
2 FROM dbo.Boehmisch_team AS tt
3 INNER JOIN dbo.Boehmisch_spieler AS sp
4 ON sp.spielernr=tt.kapitaen_nr
5 INNER JOIN dbo.Boehmisch_strafe AS st
6 ON tt.kapitaen_nr=st.spielernr
7 WHERE tt.kapitaen_nr IS NOT NULL
8 GROUP BY tt.name, sp.name, sp.vorname
```

Übungsaufgabe 15

Wie hoch ist die Gesamtstrafe für die Kapitäne der jeweiligen Teams?

Ergebnisse		Meldungen		
	team	Spieler_Name	Spieler_Vorname	Strafe
1	Bären	Bäcker	Egon	70.00
2	Llamas	Maurer	Doris	174.00
3	Tiger	Kohl	Dagmar	107.50

Übungsaufgabe 16

Wie viel Preisgeld (€) haben die Llamas bisher bekommen?

```
1 SELECT tt.name, sum(pp.preisgeld)
2 FROM dbo.Boehmisch_team AS tt
3 INNER JOIN dbo.Boehmisch_wettkampf AS wt
4 ON tt.id=wt.team_id
5 INNER JOIN dbo.Boehmisch_preisgeld AS pp
6 ON pp.wettkampf_id=wt.id
7 WHERE tt.name = 'Llamas'
8 GROUP BY tt.name
```

Übungsaufgabe 16

Wie viel Preisgeld (€) haben die Llamas bisher bekommen?

	name	(Kein Spaltenname)
1	Llamas	355,00



Übungsaufgabe 17

Wie heißen die Spieler, die mehr als drei Strafen erhalten haben?
Berücksichtigen Sie nur Strafen von mindestens 30€!

```
1 SELECT sp.name, sp.vorname, sp.spielernr
2 FROM dbo.Boehmisch_spieler AS sp
3 INNER JOIN dbo.Boehmisch_strafe AS st
4 ON st.spielernr=sp.spielernr
5 WHERE st.strafe >= 30
6 GROUP BY sp.name, sp.vorname, sp.spielernr
7 HAVING count(st.strafe)>3
8 |
```

Übungsaufgabe 17

Wie heißen die Spieler, die mehr als drei Strafen erhalten haben?
Berücksichtigen Sie nur Strafen von mindestens 30€!

 Ergebnisse  Meldungen		
	name	vorname
1	Böhmen	Manfred
2	Elfers	Rainer
3	Peters	Robert

Schlüsselwörter

DISTINCT

- Eliminiert doppelte Tupel

SELECT DISTINCT name FROM spieler

Die Anzahl der Tupel in der
Ergebnisrelation limitieren

MSSQL	Oracle	MySQL
TOP n	rownum	LIMIT m, n
SELECT TOP 3 name FROM spieler	SELECT name FROM spieler WHERE rownum<=3	SELECT name FROM spieler LIMIT 0, 3

Funktioniert nicht bei Ausgaben die
mit ORDER BY sortiert wurden

Unterabfragen / Subqueries

Als Unterabfrage (subquery) bezeichnet man eine SQL-Abfrage die innerhalb der WHERE-Klausel einer anderen SQL-Abfrage eingebettet ist.

Man unterscheidet:

➤ **Einfache Unterabfrage**

→ Ergebnis der Abfrage ist ein einzelner Wert.

➤ **Unterabfrage, die Menge zurück liefert**

→ Ergebnis der Abfrage ist mehr als ein Wert.

➤ **Korrelierende Unterabfrage**

→ Die Unterabfrage ist mit Korrelationsvariablen mit der übergeordneten Ebene verbunden.

Die Unterabfrage wird genau **einmal** durchgeführt.

Die Unterabfrage wird **für jedes** Tupel der übergeordneten Ebene **separat** durchgeführt.

Beispiel: (hier einfache Unterabfrage)

➤ **SELECT name FROM spieler
WHERE spieler_nr = (SELECT kapitaen_nr FROM team
WHERE id=1);**

	name
1	Kohl

Einfache Unterabfrage / Subqueries

Ist sichergestellt, dass eine Unterabfrage nur einen einzelnen Wert als Ergebnis liefert, kann die Unterabfrage innerhalb der übergeordneten Ebene wie eine Konstante verwendet werden.

- Alle bekannten Vergleichsoperatoren sind möglich

< > <= >= <> =

Beispiel:

Welcher Spieler ist zuletzt in den Verein eingetreten?

```
SELECT spielernr, name, vorname
```

```
FROM spieler
```

```
WHERE beitrtritt = (SELECT max(beitrtritt) FROM spieler);
```

Hinweise:

- **Außer bei Spalten-/Aggregatfunktionen** ist es eher selten, dass eine Abfrage **genau einen** Wert liefert (und hängt von der Struktur der Daten ab), daher ...
- ... sollte **in der Regel** besser davon ausgegangen werden, dass eine Unterabfrage eine **Ergebnismenge** statt einem einzelnen Wert liefert!

Unterabfragen / Subqueries

Anmerkungen / Besonderheiten:

- Unterabfragen können ineinander verschachtelt werden!
- Die Klauseln **DISTINCT** und **ORDER BY** dürfen beide nicht in Unterabfragen verwendet werden!
`SELECT __ FROM __ WHERE __ IN (SELECT ... ORDER BY __)`
`SELECT __ FROM __ WHERE __ IN (SELECT DISTINCT ...);`
- Der Zugriff auf Ergebnismengen einer Unterabfrage ist nur mit den vorgestellten Mengenquantoren möglich und nicht z.B. durch die von früher bekannten Aggregatfunktionen!
`SELECT __ FROM __ WHERE __ = Max(SELECT ...)`
- Unterabfragen können häufig ergebnisgleich zu einer entstprechenden Abfrage mit joins verwendet werden:
`SELECT spielernr FROM spieler
 WHERE spielernr in (SELECT spielernr FROM strafe);
 SELECT DISTINCT sp.spielernr FROM spieler sp
 JOIN strafe st ON sp.spielernr = st.spielernr;`

Unterabfragen, die Mengen zurück geben

Liefert eine Unterabfrage eine Wertemenge zurück, muss auf diese über folgende SQL-Mengenquantoren aus der übergeordneten Ebene zugegriffen werden.

Quantor Erklärung

IN, NOT IN Wert muss in Ergebnismenge enthalten sein

ALL Vergleich muss für alle Elemente der Ergebnismenge erfüllt sein.

ANY, SOME Vergleich muss für mindestens ein Element der Menge erfüllt sein

EXISTS Ausdruck wird wahr, wenn die Menge mindestens ein Element enthält

Unterabfragen, die Mengen zurück geben

Beispiel:

Welche Spieler haben noch keine Strafe erhalten

SELECT name, vorname

FROM spieler

WHERE

spielernr NOT IN (SELECT spielernr FROM strafe)

Synonyme Verwendung einiger Quantoren möglich:

x = ANY (a, b, c) ⇔ x IN (a, b, c)

x <> ALL (a, b, c) ⇔ x NOT IN (a, b, c)

Übungsaufgabe 18

Geben sie eine Liste mit allen Spielern aus, deren Spielernummer größer als die durchschnittliche Spielernummer ist.

```
1 SELECT sp.spielernr, sp.name, sp.vorname
2 FROM dbo.Boehmisch_spieler AS sp
3 WHERE sp.spielernr > (
4   SELECT AVG(sp2.spielernr)
5   FROM   dbo.Boehmisch_spieler AS sp2
6 )
7
```

Übungsaufgabe 18

Geben sie eine Liste mit allen Spielern aus, deren Spielernummer größer als die durchschnittliche Spielernummer ist.

	spielernr	name	vorname
1	57	Böhmen	Manfred
2	83	Hofmann	Philipp
3	95	Müller	Paul
4	100	Peters	Franz
5	104	Maurer	Doris
6	112	Bauer	Irene

Übungsaufgabe 19

Welcher Spieler hat bisher die höchste Gesamtstrafe bezahlt?

```
1 -- Falls nur nach einer Sache gefragt ist kann man sich mit top 1 zeit sparen
2 -- nur das erste Ergebnis wird ausgewählt
3 SELECT TOP 1 sp.spielernr, sp.name, sp.vorname, SUM(st.trafe)
4 FROM dbo.Boehmisch_spieler AS sp
5 INNER JOIN dbo.Boehm_strafe AS st
6 ON sp.spielernr=st.spielernr
7 GROUP BY sp.spielernr, sp.vorname, sp.name
8 ORDER BY SUM(st.trafe) DESC
9
```

```
1 -- Lösung mit Subquery um doppelte Aggregatsfunktion zu vermeiden
2 SELECT sp.spielernr, sp.name, sp.vorname, SUM(st.trafe) AS gesamtstrafe
3 FROM dbo.Boehmisch_spieler AS sp
4 INNER JOIN dbo.Boehmisch_strafe AS st
5 ON sp.spielernr = st.spielernr
6 GROUP BY sp.spielernr, sp.vorname, sp.name
7 HAVING SUM(st.trafe) = (
8     SELECT MAX(gesamtstrafe)
9     FROM (
10         SELECT spielernr, SUM(trafe) AS gesamtstrafe
11         FROM dbo.Boehmisch_strafe
12         GROUP BY spielernr
13     ) AS subquery
14 );
```

Übungsaufgabe 19

Welcher Spieler hat bisher die höchste Gesamtstrafe bezahlt?

	name	vorname	(Kein Spaltenname)
1	Böhmen	Manfred	261,50

Übungsaufgabe 20

Welcher Spieler hat nach den wenigsten Tagen nach seiner Geburt an einem Wettkampf teilgenommen? Für welches Team ist er gestartet und wer war der Kapitän des Teams?

```

1 SELECT sp.spielernr, sp.name AS spieler_name, sp.vorname,
2         DATEDIFF(day, sp.geboren, wt.datum) AS tage_seit_geburt,
3         tt.name AS team_name,
4         kap.name AS kapitaen_name, -- Kapitän'sname aus 'kap'
5         kap.vorname AS kapitaen_vorname
6 FROM dbo.Boehmischespieler AS sp
7 INNER JOIN dbo.Boehmischeswettkampf AS wt
8 ON sp.spielernr = wt.spielernr
9 INNER JOIN dbo.Boehmisches-team AS tt
10 ON wt.team_id = tt.id
11 -- Neuer Join zur Ermittlung des Kapitäns
12 INNER JOIN dbo.Boehmischespieler AS kap
13 ON tt.kapitaen_nr = kap.spielernr
14 -- Spieler mit der geringsten Anzahl an Tagen seit Geburt am Wettkampf
15 WHERE DATEDIFF(day, sp.geboren, wt.datum) = (
16     SELECT MIN(DATEDIFF(day, sp2.geboren, wt2.datum))
17     FROM dbo.Boehmischespieler AS sp2
18     INNER JOIN dbo.Boehmischeswettkampf AS wt2
19     ON sp2.spielernr = wt2.spielernr
20 );

```

- Wie heißt der/die Spieler bei dem/denen die Differenz in Tagen zwischen Geburt und Wettkampf minimal ist?
- Erweitern sie die Abfrage von a) so, dass das Team und der Kapitän des Teams zusätzlich angezeigt wird

Übungsaufgabe 20

Welcher Spieler hat nach den wenigsten Tagen nach seiner Geburt an einem Wettkampf teilgenommen? Für welches Team ist er gestartet und wie heißt der Kapitän des Teams?

	Kapitänname	Kapitänsvorname	Teamname	Spielername	Spielervorname
1	Maurer	Doris	Llamas	Bäcker	Egon

Korrelierte Unterabfragen → Probleme

Beispiel:

Geben Sie diejenigen Spieler aus, die in ihrer jeweiligen Stadt jeweils zuletzt dem Verein beigetreten sind!

Problem:

Es reicht nicht, für alle Städte das letzte Beitrittsdatum zu bestimmen, da dabei die eindeutige Zuordnung zu den restlichen Spielerdaten verloren geht:

```
SELECT ort, max(beitritt)
FROM spieler
GROUP BY ort;
```

	ort	letzter Beitritt
1	Eislingen/Fils	01.06.2006 00:00:00
2	Faurndau	01.02.2005 00:00:00
3	Göppingen	01.09.2006 00:00:00
4	Jebenhausen	01.03.1989 00:00:00
5	Rechberghausen	01.11.2004 00:00:00
6	UHINGEN	01.06.2006 00:00:00

Welche
Spieler sind
das?

Zur Lösung müsste man:

1. Alle Spieler einzeln durchlaufen.
2. Zu dem jeweiligen Ort jedes einzelnen Spielers das entsprechend letzte Beitrittsdatum ermitteln.
(→ d.h. eine extra Abfrage pro Spieler!)
3. Den Spieler nur dann in die Ergebnisrelation aufnehmen, wenn er an dem in 2. ermittelten Datum beigetreten ist.

Doch wie geht das?

Korrelierte Unterabfragen → Lösung

Beispiel:

Geben Sie diejenigen Spieler aus, die in ihrer jeweiligen Stadt jeweils zuletzt dem Verein beigetreten sind!

```
SELECT spiellernr, name, vorname, ort, beitrtritt
FROM spieler s1
WHERE beitrtritt =
    (SELECT max(beitrtritt)
     FROM spieler s2
     WHERE s2.ort=s1.ort
    );
```

Bedingung der innere Abfrage
bezieht sich auf Werte der
äußeren Abfrage!

	spiellernr	name	vorname	ort	beitritt
1	8	Neuhaus	Berta	Utingen	01.06.2006 00:00:00
2	104	Maurer	Doris	Rechberghausen	01.11.2004 00:00:00
3	28	Kohl	Claudia	Jebenhausen	01.03.1989 00:00:00
4	57	Böhmen	Manfred	Göppingen	01.09.2006 00:00:00
5	95	Müller	Paul	Faurndau	01.02.2005 00:00:00
6	112	Bauer	Irene	Eislingen/Fils	01.06.2006 00:00:00

Besonderheiten: (gegenüber einer statischen Unterabfrage)

- Unterabfrage lässt sich nicht eigenständig ausführen!
- Unterabfrage wird (fiktiv) für jedes Tupel der Oberabfrage erneut ausgeführt!

Korrelierte Unterabfragen → Detailliert

SELECT name, vorname, ort, beitrirt
FROM spieler s1 WHERE beitrirt=

SELECT MAX(beitritt) FROM spieler
s2 WHERE s2.ort=s1.ort

	name	vorname	ort	beitritt
1	Elfers	Rainer	Göppingen	2003-05-01 00:00:00.000
2	Peters	Robert	Göppingen	2002-05-03 00:00:00.000
3	Wiegand	Günther	Göppingen	1995-12-05 00:00:00.000
4	Neuhaus	Berta	Uhingen	2006-06-01 00:00:00.000
5	Kohl	Dagmar	Rechberghausen	1990-08-01 00:00:00.000
6	Kohl	Claudia	Jebenhausen	1989-03-01 00:00:00.000
7	Bischof	Dennis	Göppingen	1985-11-01 00:00:00.000
8	Bäcker	Egon	Uhingen	2000-04-01 00:00:00.000
9	Böhmen	Manfred	Göppingen	2006-09-01 00:00:00.000
10	Hofmann	Philipp	Göppingen	2000-02-01 00:00:00.000
11	Müller	Paul	Faumdau	2005-02-01 00:00:00.000
12	Peters	Franz	Göppingen	2002-05-01 00:00:00.000
13	Maurer	Doris	Rechberghausen	2004-11-01 00:00:00.000
14	Bauer	Irene	Eislingen/Fils	2006-06-01 00:00:00.000

Göppingen	2006-09-01 00:00:00.000
Uhingen	2006-06-01 00:00:00.000

	name	vorname	ort	beitritt
1	Neuhaus	Berta	Uhingen	2006-06-01 00:00:00.000
2	Maurer	Doris	Rechberghausen	2004-11-01 00:00:00.000
3	Kohl	Claudia	Jebenhausen	1989-03-01 00:00:00.000
4	Böhmen	Manfred	Göppingen	2006-09-01 00:00:00.000
5	Müller	Paul	Faumdau	2005-02-01 00:00:00.000
6	Bauer	Irene	Eislingen/Fils	2006-06-01 00:00:00.000

Korrelierte Unterabfrage → Auf einen Blick

Korrelierte Unterabfragen entstehen, wenn in der Unterabfrage explizit auf Attribute der übergeordneten Abfrage Bezug genommen wird, z.B. durch Korrelationsvariablen.

- Erkennbar daran, dass sich die Unterabfrage losgelöst von der Hauptabfrage nicht autonom ausführen ließe.
- Mehrfache (fiktive) Ausführungen der Unterabfrage für jedes Tupel der übergeordneten Abfrage.

Wissenswert:

- Korrelierte Unterabfragen können auch in der HAVING-Klausel der übergeordneten Abfrage verwendet werden, ebenso wie statische Unterabfragen, beziehen sich dann jedoch auf die jeweilige Gruppierung.

Synonyme Bezeichnungen:

- Korrelierende Unterabfrage
- Bedingte Unterabfrage
- Synchronisierte Unterabfrage

Übungsaufgabe 21

Ermitteln Sie, wer für welches Geschlecht an den meisten Wettkämpfen teilgenommen hat.

```
1  -- Das Ergebnis der äußeren Query zeigt die Anzahl der Wettkämpfe pro Spieler
2  SELECT sp.geschlecht AS 'geschlecht', sp.name AS 'name', sp.vorname AS 'vorname', count(wt.spielernr) AS 'Anzahl Spiele'
3  FROM dbo.Boehmisch_spieler AS sp
4  INNER JOIN dbo.Boehmisch_wettkampf AS wt
5  ON sp.spielernr = wt.spielernr
6  GROUP BY sp.geschlecht, sp.name, sp.vorname
7  -- Having-Klausel filtert Spieler mit maximalen Anzahl Wettkämpfe pro Geschlecht
8  HAVING COUNT(wt.spielernr) = (
9      -- Subquery ermittelt für jedes Geschlecht maximale Anzahl Wettkämpfe mit MAX(Anzahl_Spiele)
10     SELECT MAX(Anzahl_Spiele)
11     FROM (
12         -- Innerhalb Subquery werden Wettkämpfe gezählt und nach geschlecht gruppiert
13         SELECT sp2.geschlecht, sp2.name, sp2.vorname, COUNT(wt2.spielernr) AS Anzahl_Spiele
14         FROM dbo.Boehmisch_spieler AS sp2
15         INNER JOIN dbo.Boehmisch_wettkampf AS wt2
16         ON sp2.spielernr = wt2.spielernr
17         -- hierdurch durch diese WHERE-Clausel wird sichergestellt, dass sich MAX(Anzahl_Spiele)
18         -- nicht auf beide Geschlechter gleichzeitig (m,w) bezieht, sondern immer nur jeweils separat
19         -- BSP: sp2.geschlecht='m' und sp.geschlecht='m' -> da sp2.geschlecht=sp.geschlecht wird das MAX(Anzahl_spiele)
20         --      nur auf 'm' (Männer) angewendet und nicht auf die 'w' Frauen
21         --      -> Frauen werden nur mit Frauen und Männer nur mit Männern auf die maximale Anzahl der Spiele verglichen
22         --      -> Subquery wird nur für Spieler des gleichen Geschlechts ausgeführt
23         WHERE sp2.geschlecht = sp.geschlecht -- <-- das ist die entscheidende Zeile!
24         GROUP BY sp2.geschlecht, sp2.name, sp2.vorname
25     ) AS Subquery
26 );
27
```

Übungsaufgabe 21

Ermitteln Sie, wer für welches Geschlecht an den meisten Wettkämpfen teilgenommen hat.

	geschlecht	name	vorname	Anzahl Spiele
1	W	Kohl	Claudia	27
2	M	Müller	Paul	34

Übungsaufgabe 22

Erstellen sie eine Jahresliste mit dem Team, das am häufigsten im jeweiligen Jahr gewonnen hat. Und zeigen Sie auch, wie viele Spiele dieses Team gewonnen hat.

```
1 SELECT DISTINCT year(wt.datum) AS 'Jahr',  
2     tt.name AS 'Siegerteam',  
3     COUNT(*) AS 'Gewonnene Spiele'  
4 FROM dbo.Boehmisch_wettkampf AS wt  
5 INNER JOIN dbo.Boehm_team AS tt  
6 ON tt.id = wt.team_id  
7 -- Berücksichtigt nur Spiele, in denen die Punktzahl des Spielers höher war als die des Gegners (Sieg)  
8 WHERE wt.punktzahlspieler > wt.punktzahlgegner  
9 GROUP BY YEAR(wt.datum), tt.id, tt.name  
10 -- Filtert Teams, die die höchste Anzahl an Siegen pro Jahr haben  
11 HAVING COUNT(*) = (  
12     -- Subquery: Bestimmt das Team mit den meisten Siegen pro Jahr  
13     SELECT TOP 1 COUNT(*)  
14     FROM dbo.Boehmisch_wettkampf AS wt2  
15     -- Betrachtet nur Spiele im gleichen Jahr wie in der Hauptabfrage wo es einen Sieg gab  
16     WHERE year(wt2.datum) = year(wt.datum)  
17     -- durch das AND werden also verlorene und unentschiedene Spiele nicht betrachtet  
18     AND wt2.punktzahlspieler > wt2.punktzahlgegner  
19 -- Gruppiert erneut nach Team (für die Anzahl der Siege pro Team im Jahr)  
20 -- (Gruppieren nach Jahr nicht mehr notwendig, da WHERE Bedingung bereits nach Jahr filtert)  
21 GROUP BY wt2.team_id  
22 -- Gibt das Team mit den meisten Siegen (höchster COUNT) zurück  
23 ORDER BY COUNT(*) DESC  
24 )  
25 ORDER BY year(wt.datum);  
26
```

Übungsaufgabe 22

Erstellen sie eine Jahresliste mit dem Team, das am häufigsten im jeweiligen Jahr gewonnen hat. Und zeigen Sie auch, wie viele Spiele dieses Team gewonnen hat.

	Jahr	Siegerteam	Gewonnene Spiele
1	1986	Tiger	1
2	1988	Llamas	1
3	1988	Bären	1
4	1992	Bären	1
5	1993	Llamas	2
6	1994	Llamas	1
7	1995	Bären	2
8	1996	Llamas	2
9	1998	Llamas	1
10	1998	Bären	1
11	1999	Bären	1
12	1999	Llamas	1
13	2000	Llamas	5
14	2001	Tiger	2
15	2002	Llamas	3
16	2002	Bären	3
17	2003	Tiger	4
18	2003	Bären	4
19	2004	Bären	6
20	2005	Tiger	6
21	2006	Tiger	8
22	2007	Bären	11
23	2008	Llamas	9
24	2009	Llamas	12

Übungsaufgabe 23

Welche Spieler haben mehr Strafen (€) bekommen als die anderen Spieler im Durchschnitt?

```
1 SELECT sp.spielernr, sp.name, sp.vorname, SUM(st.strafe) AS gesamtstrafe
2 FROM dbo.Boehmisch_spieler AS sp
3 INNER JOIN dbo.Boehmisch_strafe AS st
4 ON sp.spielernr = st.spielernr
5 GROUP BY sp.spielernr, sp.name, sp.vorname
6 HAVING SUM(st.strafe) > (
7     -- Subquery berechnet den Durchschnitt der Gesamtstrafen aller Spieler
8     SELECT AVG(gesamtstrafe2)
9     FROM (
10         -- Innere Subquery berechnet die Gesamtsumme der Strafe pro Spieler (pro spielernr)
11         SELECT SUM(st2.strafe) AS gesamtstrafe2
12         FROM dbo.Boehmisch_strafe AS st2
13         -- Ohne Group By würde die Summe insgesamt berechnet werden und nicht pro einzelnen Spieler
14         GROUP BY st2.spielernr
15     ) AS subquery
16 )
17 ORDER BY gesamtstrafe DESC;
```

Übungsaufgabe 23

Welche Spieler haben mehr Strafen (€) bekommen als die anderen Spieler im Durchschnitt?

Ergebnisse		Meldungen		
	spielernr	name	vorname	Gesamtstrafe
1	57	Böhmen	Manfred	426.00
2	2	Elfers	Rainer	245.00
3	6	Peters	Robert	188.50
4	104	Maurer	Doris	174.00
5	8	Neuhaus	Berta	173.00

Übungsaufgabe 24

Welche Spieler haben in welchem Team mehr Spiele gewonnen als ihr Teamdurchschnitt?

```
1  SELECT
2      tt.name, sp.name, sp.vorname
3  FROM
4      dbo.Boehmisch_wettkampf AS wt
5      INNER JOIN dbo.Boehmisch_spieler AS sp
6      ON sp.spielernr = wt.spielernr
7      INNER JOIN dbo.Boehmisch_team AS tt
8      ON wt.team_id = tt.id
9  WHERE (
10     (
11         -- Zählt die Anzahl der gewonnenen Spiele pro Spieler innerhalb des Teams
12         SELECT COUNT(*)
13         FROM dbo.Boehmisch_wettkampf AS wt2
14         WHERE wt2.punktzahlspieler > wt2.punktzahlgegner -- Nur gewonnene Spiele zählen
15         AND wt2.spielernr = wt.spielernr -- Nur Spiele des aktuellen Spielers berücksichtigen
16         AND wt2.team_id = wt.team_id -- Und nur Spiele innerhalb desselben Teams betrachten
17     ) >
18     (
19         -- Berechnet den Durchschnitt der gewonnenen Spiele pro Spieler innerhalb des Teams
20         (
21             -- Zählt die Anzahl der gewonnenen Spiele des gesamten Teams
22             SELECT COUNT(*)
23             FROM dbo.Boehmisch_wettkampf AS wt2
24             WHERE wt2.punktzahlspieler > wt2.punktzahlgegner -- Nur gewonnene Spiele zählen
25             AND wt2.team_id = wt.team_id -- Nur Spiele dieses Teams zählen
26         ) / -- für den Durchschnitt geteilt rechnen: gewonnene Spiele gesamt/Anzahl Spieler
27         (
28             -- Zählt die Anzahl der einzigartigen Spieler im Team, die am Wettkampf teilgenommen haben
29             SELECT COUNT(DISTINCT wt4.spielernr)
30             FROM dbo.Boehmisch_wettkampf AS wt4
31             WHERE wt4.team_id = wt.team_id -- Spieler innerhalb desselben Teams zählen
32         )
33     )
34 )
35 -- Gruppiert die Ergebnisse nach Teamname, Spielernamen und Vorname (vermeidet Duplikate)
36 GROUP BY tt.name, sp.name, sp.vorname
37 ORDER BY tt.name, sp.name, sp.vorname;
```

Übungsaufgabe 24

Welche Spieler haben in welchem Team mehr Spiele gewonnen als ihr Teamdurchschnitt?

	name	name	vorname
1	Bären	Bauer	Irene
2	Bären	Bischof	Dennis
3	Bären	Böhmen	Manfred
4	Bären	Kohl	Claudia
5	Bären	Kohl	Dagmar
6	Bären	Maurer	Doris
7	Bären	Müller	Paul
8	Bären	Peters	Robert
9	Bären	Wiegand	Günther
10	Llamas	Bäcker	Egon
11	Llamas	Bischof	Dennis
12	Llamas	Böhmen	Manfred
13	Llamas	Hofmann	Philipp
14	Llamas	Maurer	Doris
15	Llamas	Müller	Paul
16	Llamas	Peters	Franz
17	Llamas	Peters	Robert
18	Tiger	Böhmen	Manfred
19	Tiger	Hofmann	Philipp
20	Tiger	Kohl	Dagmar
21	Tiger	Maurer	Doris
22	Tiger	Müller	Paul
23	Tiger	Peters	Franz
24	Tiger	Peters	Robert

Übungsaufgabe 25

Wie heißen die Mitglieder, die dem Verein mehr als zehn Jahre angehören, bisher mehr Wettkämpfe gewonnen als verloren haben und dabei noch keine Strafe für „Nicht Erscheinen“ bekommen haben.

Übungsaufgabe 25

Wie heißen die Mitglieder, die dem Verein mehr als zehn Jahre angehören, bisher mehr Wettkämpfe gewonnen als verloren haben und dabei noch keine Strafe für „Nicht Erscheinen“ bekommen haben.

	name	vomame
1	Peters	Robert
2	Bischof	Dennis

Übung 26 Outer Join

Geben Sie eine nach Datum absteigend sortierte Wettkampfliste aller Spieler aus und ergänzen diese ggf. um ein erhaltenes Preisgeld. Sollte es bereits ausgezahlt worden sein, geben Sie das entsprechende Datum dazu an.

Ergänzen Sie diese Liste dann um Spieler-Strafen, soweit es diese zu diesem Wettkampf gegeben hat.

	Wettkampf am	TEAM	spielernr	name	vorname	preisgeld	ausgezahlt_am	strafe	spielernr	Strafe Datum
91	2008-05-25 00:00:00.000	Bären	112	Bauer	Irene	NULL	NULL	NULL	NULL	NULL
92	2008-05-22 00:00:00.000	Bären	83	Hofmann	Philipp	NULL	NULL	NULL	NULL	NULL
93	2008-05-13 00:00:00.000	Tiger	57	Böhmen	Manfred	NULL	NULL	NULL	NULL	NULL
94	2008-05-08 00:00:00.000	Llamas	7	Wiegand	Günther	NULL	NULL	NULL	NULL	NULL
95	2008-05-02 00:00:00.000	Llamas	112	Bauer	Irene	NULL	NULL	14,50	39	2008-05-02 00:00:00.000
96	2008-05-01 00:00:00.000	Llamas	57	Böhmen	Manfred	NULL	NULL	28,00	57	2008-05-01 00:00:00.000
97	2008-04-29 00:00:00.000	Llamas	104	Maurer	Doris	NULL	NULL	NULL	NULL	NULL
98	2008-04-26 00:00:00.000	Bären	28	Kohl	Claudia	NULL	NULL	NULL	NULL	NULL
99	2008-04-22 00:00:00.000	Llamas	95	Müller	Paul	NULL	NULL	NULL	NULL	NULL
100	2008-04-06 00:00:00.000	Llamas	57	Böhmen	Manfred	NULL	NULL	NULL	NULL	NULL
101	2008-03-29 00:00:00.000	Tiger	100	Peters	Franz	NULL	NULL	NULL	NULL	NULL
102	2008-03-25 00:00:00.000	Tiger	95	Müller	Paul	NULL	NULL	NULL	NULL	NULL
103	2008-03-20 00:00:00.000	Bären	112	Bauer	Irene	NULL	NULL	NULL	NULL	NULL
104	2008-03-14 00:00:00.000	Tiger	7	Wiegand	Günther	50,00	2008-05-31 00:00:00.000	NULL	NULL	NULL
105	2008-03-11 00:00:00.000	Llamas	83	Hofmann	Philipp	NULL	NULL	NULL	NULL	NULL
106	2008-03-06 00:00:00.000	Llamas	100	Peters	Franz	35,00	2008-09-22 00:00:00.000	NULL	NULL	NULL
107	2008-02-28 00:00:00.000	Llamas	112	Bauer	Irene	NULL	NULL	NULL	NULL	NULL

(Auszug aus der Gesamtliste)

Übung 26 Outer Join

Gibt es Wettkämpfe, an denen es Preisgelder und gleichzeitig Strafen gab?

Vereinfachen und verkürzen Sie die Liste für diese Fragestellung!

	Wettkampf am	TEAM	spielernr	name	vorname	preisgeld	ausgezahlt_am	strafe	spielernr	Strafe Datum
1	2007-01-16 00:00:00.000	Bären	57	Böhmen	Manfred	55,00	2009-09-05 00:00:00.000	40,00	57	2007-01-16 00:00:00.000

```

1 SELECT
2   wt.datum, tt.name, wt.spielernr, sp.name, sp.vorname, pr.preisgeld, pr.ausgezahlt_am,
3   st.strafe, st.spielernr, st.datum
4
5 FROM       dbo.Boehmisch_wettkampf AS wt
6 -- INNER JOIN gibt nur Übereinstimmungen aus (es werden nur Spieler angezeigt, die an einem Wettkampf teilgenommen haben).
7 INNER JOIN dbo.Boehmisch_spieler AS sp
8   ON sp.spielernr=wt.spielernr
9 INNER JOIN dbo.Boehmisch_team AS tt
10  ON tt.id = wt.team_id
11
12 -- LEFT JOIN: Zeige alle Wettkämpfe, auch wenn kein Preisgeld ausgezahlt wurde.
13 -- Falls kein Preisgeld existiert, sind die Spalten pr.preisgeld und pr.ausgezahlt_am NULL.
14 LEFT JOIN  dbo.Boehmisch_preisgeld AS pr
15   ON pr.wettkampf_id = wt.id
16
17 -- LEFT JOIN: Zeige auch Wettkämpfe ohne Strafen.
18 -- Nur Strafen, die am selben Tag wie der Wettkampf vergeben wurden UND zur gleichen Spieler-Nr. gehören, werden angezeigt.
19 LEFT JOIN  dbo.Boehmisch_strafe AS st
20   ON st.datum = wt.datum
21   AND st.spielernr = wt.spielernr
22 ORDER BY wt.datum DESC;

```