Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

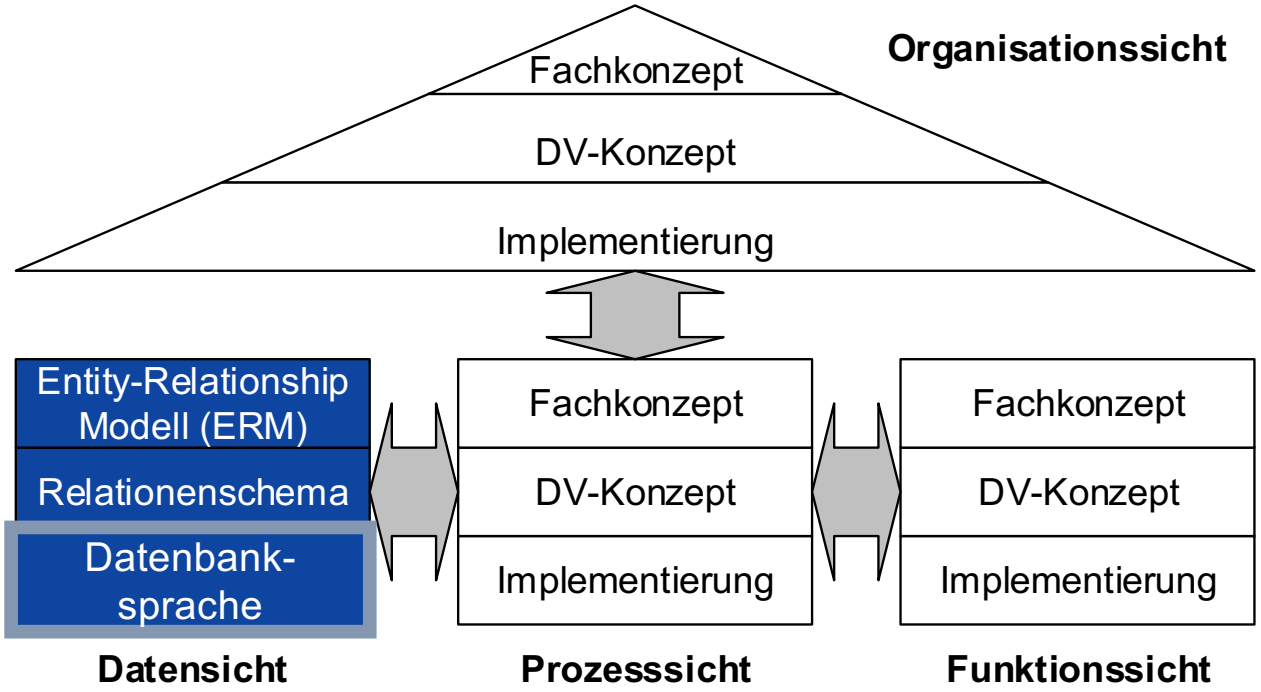
Datenmanagement & -analyse

Übung 5 – Datenbankabfragen 2

Dr. Nikolai Stein
Lehrstuhl für WI & BA
Julius-Maximilians-Universität Würzburg

Sommersemester 2021

Architektur Integrierter Informationssysteme

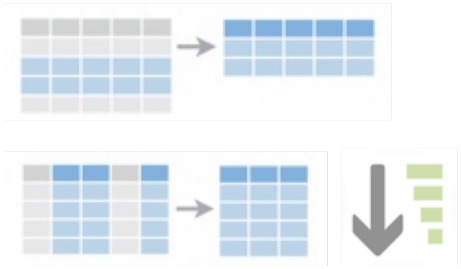


Scheer (1992)

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Die vereinfachte Welt der Abfragen

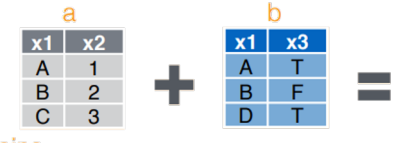
Einfache Tabellenoperationen



Gruppieren & Zusammenfassen

Fensteroperationen

Verbundoperationen



Funktionen auf einer Tabelle

Funktionen auf mehreren Tabellen

Normalisierte Daten liegen nicht in einer einzelnen Tabelle

Wenn wir uns auf relationale Datenbanken verlassen oder Daten aus verschiedenen Systemen verwenden, müssen wir Datensätze effizient kombinieren

Beispiele

- Wie können wir Verspätungen an Flughäfen auf einer Karte anzeigen?
- Hängt die Anzahl der Triebwerke mit der Geschwindigkeit zusammen?
- Sind Boeing-Flugzeuge pünktlicher als Airbus-Flugzeuge?

name	instrument
John	guitar
Paul	bass
George	guitar
Ringo	drums
Stuart	bass
Pete	drums

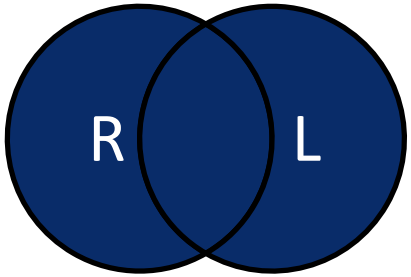
+

name	band
John	T
Paul	T
George	T
Ringo	T
Brian	F

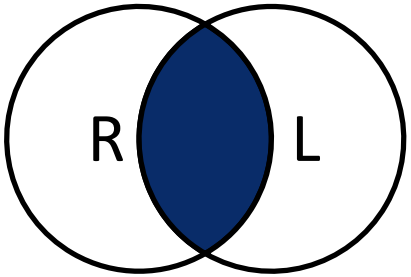
= ?

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

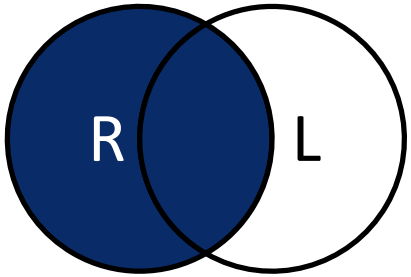
Verbundoperationen



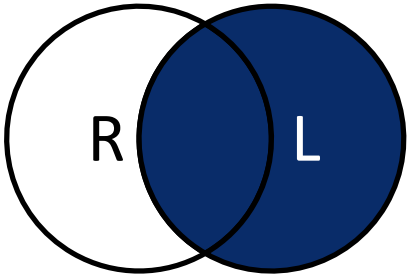
Full Outer Join



Inner Join



Left Join



Right Join

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Verbundoperationen Beispiel

ID	NAME	Telefon
1	Flath	85128
2	Griebel	86810
3	Rottmann	87967
4	Greif	83168
5	Oberdorf	88587
6	Stein	85119

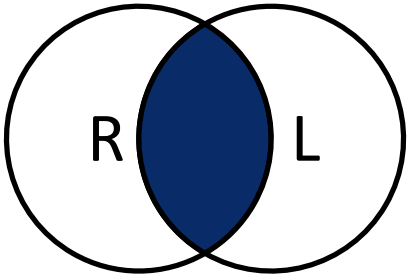
ID	Kurs	Betreuer	Level
1	AIS	1	Master
2	DSS	4	Master
3	PUE	6	Bachelor
4	DMA	1	Bachelor
5	PDS	2	Master
6	MPS	3	Bachelor

Verbundoperationen Inner Join

ID	NAME	Telefon
1	Flath	85128
2	Griebel	86810
3	Rottmann	87967
4	Greif	83168
5	Oberdorf	88587
6	Stein	85119

+

ID	Kurs	Betreuer	Level
1	AIS	1	Master
2	DSS	4	Master
3	PUE	6	Bachelor
4	DMA	1	Bachelor
5	PDS	2	Master
6	MPS	3	Bachelor



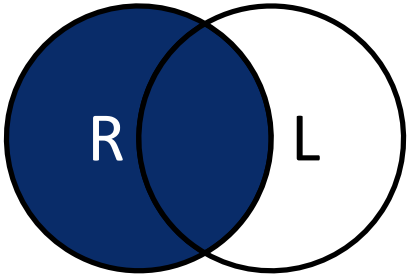
A.ID	NAME	Telefon	B.ID	Kurs	Level
1	Flath	85128	1	AIS	Master
1	Flath	85128	4	DMA	Bachelor
2	Griebel	86810	5	PDS	Master
3	Rottmann	87967	6	MPS	Bachelor
4	Greif	83168	2	DSS	Master
6	Stein	85119	3	PDS	Bachelor

Verbundoperationen Left Join

ID	NAME	Telefon
1	Flath	85128
2	Griebel	86810
3	Rottmann	87967
4	Greif	83168
5	Oberdorf	88587
6	Stein	85119

+

ID	Kurs	Betreuer	Level
1	AIS	1	Master
2	DSS	4	Master
3	PUE	6	Bachelor
4	DMA	1	Bachelor
5	PDS	2	Master
6	MPS	3	Bachelor



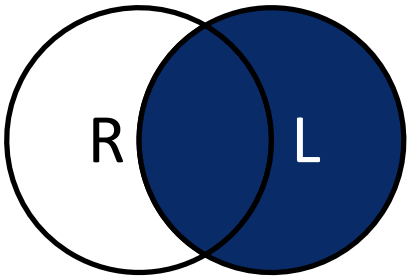
A.ID	NAME	Telefon	B.ID	Kurs	Level
1	Flath	85128	1	AIS	Master
1	Flath	85128	4	DMA	Bachelor
2	Griebel	86810	5	PDS	Master
3	Rottmann	87967	6	MPS	Bachelor
4	Greif	83168	2	DSS	Master
5	Oberdorf	88587	NA	NA	NA
6	Stein	85119	3	PDS	Bachelor

Verbundoperationen Right Join

ID	NAME	Telefon
1	Flath	85128
2	Griebel	86810
3	Rottmann	87967
4	Greif	83168
5	Oberdorf	88587
6	Stein	85119

+

ID	Kurs	Betreuer	Level
1	AIS	1	Master
2	DSS	4	Master
3	PUE	6	Bachelor
4	DMA	1	Bachelor
5	PDS	2	Master
6	MPS	3	Bachelor



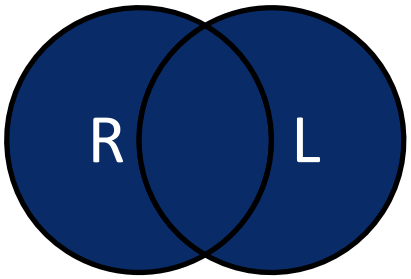
A.ID	NAME	Telefon	B.ID	Kurs	Level
1	Flath	85128	1	AIS	Master
4	Greif	83168	2	DSS	Master
6	Stein	85119	3	PUE	Bachelor
1	Flath	85128	4	DMA	Bachelor
2	Griebel	86810	5	PDS	Master
3	Rottmann	87967	6	MPS	Bachelor

Verbundoperationen Outer Join

ID	NAME	Telefon
1	Flath	85128
2	Griebel	86810
3	Rottmann	87967
4	Greif	83168
5	Oberdorf	88587
6	Stein	85119

+

ID	Kurs	Betreuer	Level
1	AIS	1	Master
2	DSS	4	Master
3	PUE	6	Bachelor
4	DMA	1	Bachelor
5	PDS	2	Master
6	MPS	3	Bachelor



A.ID	NAME	Telefon	B.ID	Kurs	Level
1	Flath	85128	1	AIS	Master
1	Flath	85128	4	DMA	Bachelor
2	Griebel	86810	5	PDS	Master
3	Rottmann	87967	6	MPS	Bachelor
4	Greif	83168	2	DSS	Master
5	Oberdorf	88587	NA	NA	NA
6	Stein	85119	3	PDS	Bachelor

Aufbau einer SQL-Abfrage

Eine SQL Abfrage kann aus den folgenden Elementen bestehen:

- SQL Keyword (z.B. SELECT, FROM)
- Spaltennamen
- Tabellennamen
- Wildcard Symbole (%, _)
- Funktionen
- Filter Kriterien

- Syntax

```
SELECT [ALL | DISTINCT]
  spaltenname1, ….
FROM tabelle1 [, tabelle2, …]
[WHERE bedingung];
```

- Beispiel
  - Zeige alle Kunden an.

```
SELECT * FROM Customers;
```

  - Zeige alle Artikel mit Preis von weniger als 50 an.

```
SELECT ProductName, Price
FROM Products
WHERE Price < 50;
```

SQL Wildcard Symbole und Funktionen

- Wildcard Symbole dienen als Platzhalter für unbekannte Zeichen.
- Werden Wildcards verwendet wird nicht mehr mittels „=„ sondern mit LIKE verglichen

- %: beliebig viele Zeichen in Zeichenkette
  - Beispiel: **'A%BC'** liefert **'**ADEFGHBC**'** oder **'**ABC**'**, …

- _: genau ein Zeichen in Zeichenkette
  - Beispiel: **'**A_BC**'** liefert **'**ADBC**'**, …

- Manche Fragen können nicht direkt auf den vorhandenen Daten beantwortet werden sondern benötigen Aggregationen
- Häufig verwendete Funktionen sind u.a.:
  - max
  - min
  - sum
  - avg
  - count
  - round
  - year, month, day, hour,…

Aufgabe 1 – SQL Island

Nach einem Flugzeugabsturz stellen Sie fest, dass Sie der einzige Überlebende sind. Sie landen auf der Insel SQL Island und das Ziel des Spiels ist es, von dieser Insel zu entkommen.

a) Welche Bewohner auf der Insel sind friedlich?

```sql
SELECT *
FROM bewohner
WHERE status = 'friedlich';
```

b) Finden Sie einen friedlichen Waffenschmied der Ihnen ein Schwert schmieden kann.

```sql
SELECT *
FROM bewohner
WHERE status = 'friedlich'
AND beruf = 'Waffenschmied';
```

Aufgabe 1 – SQL Island

Nach einem Flugzeugabsturz stellen Sie fest, dass Sie der einzige Überlebende sind. Sie landen auf der Insel SQL Island und das Ziel des Spiels ist es, von dieser Insel zu entkommen.

c) Sie haben sehr wenige friedliche Waffenschmiede gefunden. Vielleicht kann Ihnen ein anderer Schmied (z.B. Hufschmied, Schmied, Waffenschmied, etc) weiterhelfen.

```sql
SELECT *
FROM bewohner
WHERE status = 'friedlich'
AND lower(beruf) LIKE '%schmied';
```

d) Auf Ihrer Suche begegnen Sie dem Bürgermeister von Affenstadt Paul. Er bietet an Sie als Bewohner seines Dorfes einzutragen.

```sql
INSERT INTO bewohner
VALUES (20, "Niko", 1, 'm', 'wissenschaftlicher Mitarbeiter', 0, 'friedlich');
```

Aufgabe 1 – SQL Island

Nach einem Flugzeugabsturz stellen Sie fest, dass Sie der einzige Überlebende sind. Sie landen auf der Insel SQL Island und das Ziel des Spiels ist es, von dieser Insel zu entkommen.

e) Listen Sie alle Gegenstände auf die niemandem gehören.

```sql
SELECT *
FROM gegenstand
WHERE besitzer IS NULL;
```

f) Sammeln Sie die Kaffeetasse auf und tragen sich als Besitzer ein.

```sql
UPDATE gegenstand
SET besitzer = 20
WHERE gegenstand = 'Kaffeetasse';
```

g) Sammeln Sie alle Gegenstände die niemandem gehören auf.

```sql
UPDATE gegenstand
SET besitzer = 20
WHERE besitzer IS NULL;
```

Aufgabe 1 – SQL Island

Nach einem Flugzeugabsturz stellen Sie fest, dass Sie der einzige Überlebende sind. Sie landen auf der Insel SQL Island und das Ziel des Spiels ist es, von dieser Insel zu entkommen.

h) Listen Sie alle Gegenstände auf die Ihnen gehören

```sql
SELECT *
FROM gegenstand
WHERE besitzer = 20;
```

i) Eventuell möchte jemand Ihre Gegenstände kaufen. Finden Sie alle friedlichen Bewohner mit dem Beruf Haendler oder Kaufmann.

```sql
SELECT *
FROM bewohner
WHERE beruf IN ('Haendler', 'Kaufmann')
AND status = 'friedlich';
```

Aufgabe 1 – SQL Island

Nach einem Flugzeugabsturz stellen Sie fest, dass Sie der einzige Überlebende sind. Sie landen auf der Insel SQL Island und das Ziel des Spiels ist es, von dieser Insel zu entkommen.

j) Der Bewohner mit der Nummer 15 ist bereit ihnen den Ring und die Teekanne abzukaufen. Übertragen Sie das Eigentum.

```sql
UPDATE gegenstand
SET besitzer = 15
WHERE gegenstand IN ('Ring', 'Teekanne');
```

k) Sie erhalten 120 Gold für die beiden Gegenstände.

```sql
UPDATE bewohner
SET gold = gold + 120
WHERE bewohnernr = 20;
```

Aufgabe 1 – SQL Island

Nach einem Flugzeugabsturz stellen Sie fest, dass Sie der einzige Überlebende sind. Sie landen auf der Insel SQL Island und das Ziel des Spiels ist es, von dieser Insel zu entkommen.

l) Um sich etwas Gold zu verdienen beschließen Sie sich als Bäcker zu bewerben. Finden Sie alle Baecker auf der Insel und sortieren Sie diese absteigend nach ihrem Vermögen.

```sql
SELECT *
FROM bewohner
WHERE beruf = 'Baecker'
ORDER BY gold DESC;
```

m) Der reichste Bäcker stellt Sie an und kauft Ihnen 10.000 Brötchen für 100 Gold ab. Aktualisieren Sie Ihren Kontostand.

```sql
UPDATE bewohner
SET gold = gold + 100
WHERE bewohnernr = 20;
```

Aufgabe 1 – SQL Island

Nach einem Flugzeugabsturz stellen Sie fest, dass Sie der einzige Überlebende sind. Sie landen auf der Insel SQL Island und das Ziel des Spiels ist es, von dieser Insel zu entkommen.

n) Sie kaufen Sie sich für 150 Gold ein Schwert und können nun überall hin reisen.

```sql
UPDATE bewohner
SET gold = gold - 150
WHERE bewohnernr = 20;
```

```sql
INSERT INTO gegenstand
VALUES ('Schwert', 20);
```

o) Finden Sie einen Piloten auf der Insel der Sie nach Hause fliegen kann.

```sql
SELECT *
FROM bewohner
WHERE beruf = 'Pilot';
```

Aufgabe 1 – SQL Island

Nach einem Flugzeugabsturz stellen Sie fest, dass Sie der einzige Überlebende sind. Sie landen auf der Insel SQL Island und das Ziel des Spiels ist es, von dieser Insel zu entkommen.

p) Leider wird der Pilot von „Dirty Dieter“ gefangen gehalten. Finden Sie heraus in welchem Dorf der Pilot gefangen gehalten wird.

```sql
SELECT dorf.name
FROM bewohner
LEFT JOIN dorf
ON bewohner.dorfnr = dorf.dorfnr
WHERE  bewohner.name = "Dirty Dieter";
```

q) Finden Sie den Häuptling des Dorfes in dem der Pilot gefangen gehalten wird.

```sql
SELECT bewohner.name
FROM bewohner
INNER JOIN dorf
ON bewohner.bewohnernr = dorf.haeuptling
WHERE dorf.name = "Zwiebelhausen";
```

Aufgabe 1 – SQL Island

Nach einem Flugzeugabsturz stellen Sie fest, dass Sie der einzige Überlebende sind. Sie landen auf der Insel SQL Island und das Ziel des Spiels ist es, von dieser Insel zu entkommen.

r) Bevor Sie einen Angriff auf das Dorf planen möchten Sie wissen wie viele Einwohner es hat.

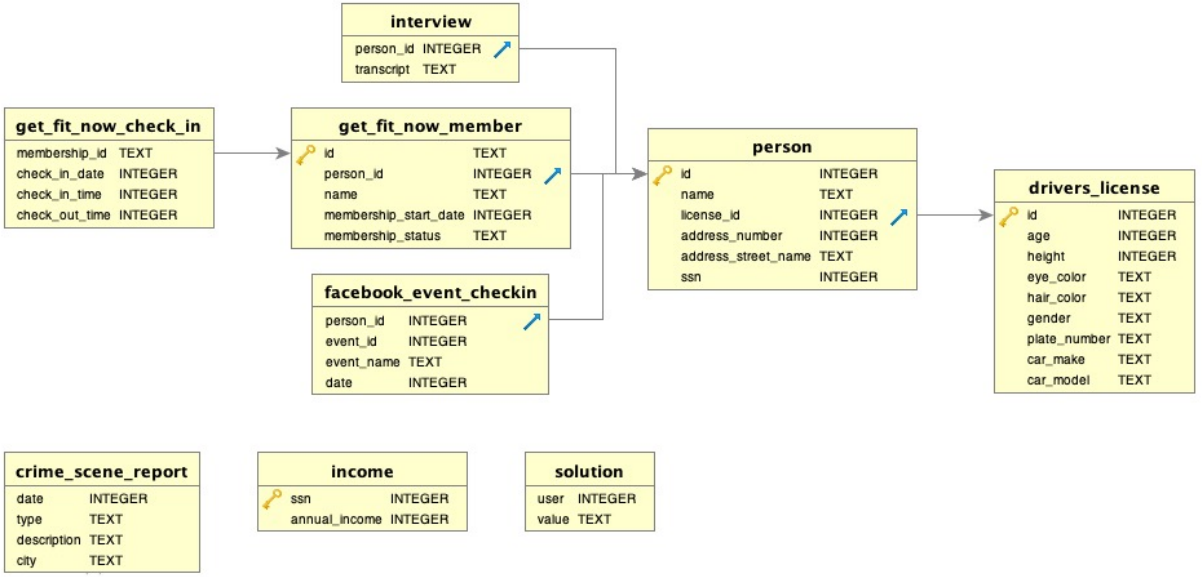
```sql
SELECT COUNT(1)
FROM bewohner
LEFT JOIN dorf
ON bewohner.dorfnr = dorf.dorfnr
WHERE dorf.name = 'Zwiebelhausen';
```

s) Ihnen gelingt es den Piloten zu befreien. Selbstverständlich ist Ihnen der Pilot nun friedlich gestimmt und fliegt Sie aus. Aktualisieren Sie seinen Status.

```sql
UPDATE bewohner
SET status = 'friedlich'
WHERE beruf = 'Pilot';
```

Aufgabe 2 – SQL Murder Mystery

In Ihrem neuen Nebenjob unterstützen Sie eine Polizeidienststelle bei der Dokumentation von Ermittlungen. Der zuständige Kommissar hat Ihnen entsprechend ein Protokoll zur Erfassung übergeben, Sie haben dieses jedoch leider verloren. Sie erinnern sich aber noch vage daran, dass es sich bei dem aufzuklärenden Verbrechen um einen Mord handelt, der sich irgendwann am 15. Januar 2018 ereignet hat und in „SQL City“ stattfand. Da Sie Zugriff auf die gesamte Polizeidatenbank haben beschließen Sie den Fall mit Hilfe Ihrer SQL-Kenntnisse selber zu lösen.



Aufgabe 2 – SQL Murder Mystery

a) Finden Sie den zugehörigen Tatortbericht

```
SELECT *
FROM crime_scene_report
WHERE date = 20180115
AND lower(city) = 'sql city'
AND type = 'murder';
```

Aufgabe 2 – SQL Murder Mystery

b) Finden Sie den ersten Zeugen

```
SELECT *, max(address_number)
FROM person
WHERE lower(address_street_name) = 'northwestern dr'
GROUP BY address_street_name
```

Aufgabe 2 – SQL Murder Mystery

c) Identifizieren Sie den zweiten Zeugen

```
SELECT *
FROM person
WHERE lower(address_street_name) = 'franklin ave'
AND name LIKE 'Annabel %';
```

Aufgabe 2 – SQL Murder Mystery

d) Finden Sie die Interviewprotokolle der beiden Zeugen

```
SELECT *
FROM person
LEFT JOIN interview
ON person.id = interview.person_id
WHERE id IN (14887, 16371);
```

Aufgabe 2 – SQL Murder Mystery

e) Werten Sie die Protokolle aus und finden Sie den Mörder

Zeuge 1:

```sql
SELECT *
FROM person
LEFT JOIN get_fit_now_member
ON person.id = get_fit_now_member.person_id
INNER JOIN drivers_license
ON person.license_id = drivers_license.id
WHERE get_fit_now_member.id LIKE '48Z%'
AND membership_status = 'gold';
```

Zeuge 2:

```sql
SELECT *
FROM person
LEFT JOIN get_fit_now_member
ON person.id = get_fit_now_member.person_id
LEFT JOIN get_fit_now_check_in
ON get_fit_now_member.id = get_fit_now_check_in.membership_id
WHERE get_fit_now_check_in.check_in_date = 20180109
AND get_fit_now_check_in.check_in_time <= 1700
AND get_fit_now_check_in.check_out_time >= 1600
AND person.id != 16371;
```

Aufgabe 2 – SQL Murder Mystery

f) Wer war der Auftraggeber des Mörders?

```sql
SELECT *
FROM person
LEFT JOIN interview
ON person.id = interview.person_id
WHERE id = 67318;
```

```sql
SELECT *, COUNT(1) AS n_besuche
FROM facebook_event_checkin
LEFT JOIN person
ON facebook_event_checkin.person_id = person.id
INNER JOIN drivers_license
ON person.license_id = drivers_license.id
WHERE lower(event_name) = 'sql symphony concert'
AND DATE >= 20171201 AND DATE <= 20171231
GROUP BY person_id
HAVING n_besuche >= 3;
```

Aufgabe 2 – SQL Murder Mystery

```sql
WITH zeuge1 AS (
  SELECT *, max(address_number)
  FROM person
  WHERE lower(address_street_name) = 'northwestern dr'
  GROUP BY address_street_name
  ),
  zeuge2 AS (
   SELECT *
   FROM person
   WHERE lower(address_street_name) = 'franklin ave'
   AND name LIKE 'Annabel %'
  ),
  zeugen AS (
    SELECT id, 1 AS zeuge FROM zeuge1
    UNION
    SELECT id, 2 AS zeuge FROM zeuge2
    )
SELECT *
FROM zeugen
LEFT JOIN interview
ON zeugen.id = interview.person_id;
```

```sql
WITH fitnessstudio_checkins AS (
  SELECT person_id, name
  FROM get_fit_now_member
  LEFT JOIN get_fit_now_check_in
  ON get_fit_now_member.id = get_fit_now_check_in.membership_id
  WHERE membership_status = 'gold'
  AND id LIKE '48Z%'
  AND check_in_date = 20180109
  ),
  verdaechtige AS (
    SELECT fitnessstudio_checkins.person_id,
    fitnessstudio_checkins.name, plate_number, gender
    FROM fitnessstudio_checkins
    LEFT JOIN person
    ON fitnessstudio_checkins.person_id = person.id
    INNER JOIN drivers_license
    ON person.license_id = drivers_license.id
    WHERE plate_number LIKE '%H42W%'
    AND gender = 'male'
   )

 SELECT *
 FROM verdaechtige;
```