https://programmer.help/blogs/fifty-sql-exercise-questions-and-answers-and-detailed-analysis.html

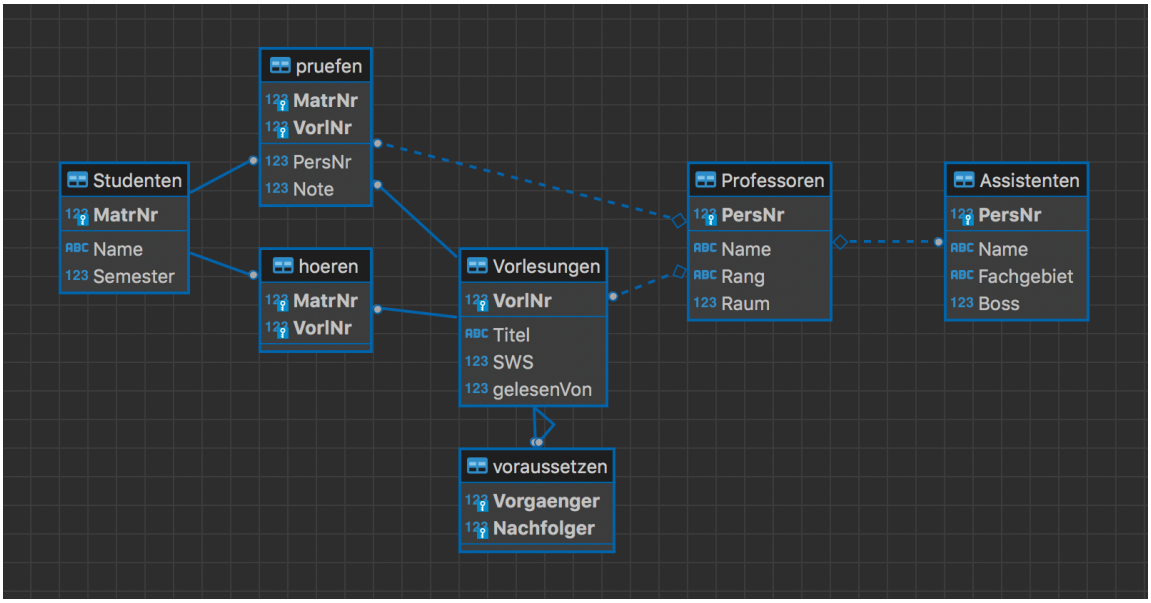
In [1]:
```
# Hier ist nur Code zum Initialisieren der Umgebeung, bitte gehen Sie weiter

# Keine langen Fehlermeldungen
import sys
ipython = get_ipython()

def exception_handler(exception_type, exception, traceback):
    print("%s: %s" % (exception_type.__name__, exception), file=sys.stderr)

ipython._showtraceback = exception_handler

# Lade die Erweiterung, damit wir SQL Befehle nutzen können
%load_ext sql

# Verbinde Dich zu einer in - Memory Datenbank
%sql sqlite:///uni.db
%sql PRAGMA foreign_keys = ON
```

```
 * sqlite:///uni.db
Done.
```

Out[1]:
```
[]
```

In [2]:
```
!echo .schema | sqlite3 uni.db
```

```
CREATE TABLE Studenten
       (MatrNr          INTEGER PRIMARY KEY,
        Name            VARCHAR(30) NOT NULL,
        Semester        INTEGER);
CREATE TABLE Professoren
       (PersNr          INTEGER PRIMARY KEY,
        Name            VARCHAR(30) NOT NULL,
        Rang            CHAR(2) CHECK (Rang in ('C2', 'C3', 'C4')),
        Raum            INTEGER UNIQUE);
CREATE TABLE Assistenten
       (PersNr          INTEGER PRIMARY KEY,
        Name            VARCHAR(30) NOT NULL,
        Fachgebiet      VARCHAR(30),
        Boss            INTEGER,
        FOREIGN KEY     (Boss) REFERENCES Professoren);
CREATE TABLE Vorlesungen
       (VorlNr          INTEGER PRIMARY KEY,
        Titel           VARCHAR(30),
        SWS             INTEGER,
        gelesenVon      INTEGER REFERENCES Professoren);
CREATE TABLE hoeren
       (MatrNr          INTEGER REFERENCES Studenten ON DELETE CASCADE,
        VorlNr          INTEGER REFERENCES Vorlesungen ON DELETE CASCADE,
        PRIMARY KEY     (MatrNr, VorlNr));
CREATE TABLE voraussetzen
       (Vorgaenger      INTEGER REFERENCES Vorlesungen ON DELETE CASCADE,
        Nachfolger      INTEGER REFERENCES Vorlesungen ON DELETE NO ACTION,
        PRIMARY KEY     (Vorgaenger, Nachfolger));
CREATE TABLE pruefen
       (MatrNr          INTEGER REFERENCES Studenten ON DELETE CASCADE,
        VorlNr          INTEGER REFERENCES Vorlesungen,
        PersNr          INTEGER REFERENCES Professoren,
        Note            NUMERIC(2,1) CHECK (Note between 0.7 and 5.0),
        PRIMARY KEY     (MatrNr, VorlNr));
```



Aufgabe 1

Welche Studenten haben in der Prüfung der Vorlesung "Logik" mehr Punkte als in der Vorlesung "Ethik"

In [20]:
```
%%sql

SELECT s.*,
s.MatrNr,
(SELECT Note
     FROM pruefen p
     JOIN Vorlesungen v USING(VorlNr)
     WHERE p.MatrNr = s.MatrNr AND v.Titel = "Logik"
) as logik,
(SELECT Note
     FROM pruefen p
     JOIN Vorlesungen v USING(VorlNr)
     WHERE p.MatrNr = s.MatrNr AND v.Titel = "Ethik"
) as ethik
FROM Studenten as s
WHERE ethik IS NOT NULL AND logik IS NOT NULL;
```

```
 * sqlite:///uni.db
Done.
```

Out[20]:

MatrNr	Name	Semester	MatrNr_1	logik	ethik
25403	Jonas	12	25403	4	2
26830	Aristoxenos	8	26830	2	1
28106	Carnap	3	28106	3	4
29555	Feuerbach	2	29555	1	1

In [21]:
```
%%sql

SELECT s.*,
    (SELECT Note
     FROM pruefen p
     JOIN Vorlesungen v USING(VorlNr)
     WHERE p.MatrNr = s.MatrNr AND v.Titel = "Logik"
    ) as logik,
    (SELECT Note
     FROM pruefen p
     JOIN Vorlesungen v USING(VorlNr)
     WHERE p.MatrNr = s.MatrNr AND v.Titel = "Ethik"
    ) as ethik
FROM Studenten s
WHERE ethik > logik;
```

```
 * sqlite:///uni.db
Done.
```

Out[21]:

MatrNr	Name	Semester	logik	ethik
28106	Carnap	3	3	4

In [45]:
```
%%sql

WITH ethik(MatrNr, Note) AS (
    SELECT p.MatrNr, p.Note
    FROM pruefen p
    JOIN Vorlesungen v ON(p.VorlNr = v.VorlNr)
    WHERE v.Titel = "Ethik"
), logik(MatrNr, Note) AS (
    SELECT p.MatrNr, p.Note
    FROM pruefen p
    JOIN Vorlesungen v ON(p.VorlNr = v.VorlNr)
```

```
    WHERE v.Titel = "Logik"
)
SELECT s.*, logik.Note as logik, ethik.Note as ethik
FROM Studenten s
JOIN logik USING(MatrNr)
JOIN ethik USING(MatrNr)
WHERE ethik > logik;
```

```
 * sqlite:///uni.db
Done.
```

Out[45]:

MatrNr	Name	Semester	logik	ethik
28106	Carnap	3	3	4

In [54]:

```
%%sql

WITH ethik(MatrNr, Note) AS (
    SELECT p.MatrNr, p.Note
    FROM pruefen p
    JOIN Vorlesungen v ON(p.VorlNr = v.VorlNr)
    WHERE v.Titel = "Ethik"
), logik(MatrNr, Note) AS (
    SELECT p.MatrNr, p.Note
    FROM pruefen p
    JOIN Vorlesungen v ON(p.VorlNr = v.VorlNr)
    WHERE v.Titel = "Logik"
)
SELECT *
FROM Studenten s
WHERE s.MatrNr IN (
    SELECT s.MatrNr
    FROM Studenten s
    JOIN logik USING(MatrNr)
    JOIN ethik USING(MatrNr)
    WHERE ethik.Note > logik.Note
)
```

```
 * sqlite:///uni.db
Done.
```

Out[54]:

MatrNr	Name	Semester
28106	Carnap	3

Aufgabe 2

Geben Sie Matrikelnummer, name und durchschnittsnote von Studenten aus, deren Durschnittsnote besser als 3 ist.

In [69]:

```
%%sql
SELECT p.MatrNr, AVG(p.Note) as durchschnitt, GROUP_CONCAT(p.Note) as noten
GROUP BY p.MatrNr
HAVING durchschnitt < 3;
```

```
 * sqlite:///uni.db
Done.
```

Out[69]:

MatrNr	durchschnitt	noten
26830	1.5	2,1
27550	2.0	2
28106	2.6666666666666665	3,1,4
29555	1.0	1,1

In [70]:
```
%%sql
SELECT p.MatrNr, AVG(p.Note) as durchschnitt, GROUP_CONCAT(p.Note) as noten
WHERE p.Note < 3
GROUP BY p.MatrNr;
```

```
 * sqlite:///uni.db
Done.
```

Out[70]:

MatrNr	durchschnitt	noten
25403	2.0	2
26830	1.5	2,1
27550	2.0	2
28106	1.0	1
29555	1.0	1,1

Aufgabe 3

Welche Studenten haben in welchem Fach mit welcher Note bereits an mindestens einer Prüfung teilgenommen

In [7]:
```
%%sql

SELECT *
FROM Studenten s
JOIN pruefen USING(MatrNr)
```

```
 * sqlite:///uni.db
Done.
```

Out[7]:

MatrNr	Name	Semester	VorlNr	PersNr	Note
28106	Carnap	3	5001	2126	1
25403	Jonas	12	5041	2125	2
27550	Schopenhauer	6	4630	2137	2
25403	Jonas	12	4052	2125	4
26830	Aristoxenos	8	4052	2125	2
28106	Carnap	3	4052	2125	3
29555	Feuerbach	2	4052	2125	1
26830	Aristoxenos	8	5041	2125	1
28106	Carnap	3	5041	2125	4
29555	Feuerbach	2	5041	2125	1

Aufgabe 4

Welche Studenten haben schon Prüfungen gesammelt?

In [9]:
```
%%sql

SELECT DISTINCT s.*
FROM Studenten s
JOIN pruefen USING(MatrNr)
```

```
 * sqlite:///uni.db
Done.
```

Out[9]:

MatrNr	Name	Semester
25403	Jonas	12
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29555	Feuerbach	2

In [10]:
```
%%sql

SELECT s.*
FROM Studenten s
WHERE MatrNr IN (SELECT MatrNr FROM pruefen)
```

```
 * sqlite:///uni.db
Done.
```

Out[10]:

MatrNr	Name	Semester
25403	Jonas	12
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29555	Feuerbach	2

Aufgabe 5

Geben Sie alle Informationen zu jedem Studierenden aus. Darüber hinaus geben Sie noch die Anzahl der aktuell gehörten Kurse, die Anzahl der bereits gehörten Kurse und die durschnittsnote aus.

In [33]:
```
%%sql

SELECT s.*,
    (SELECT COUNT(h.MatrNr) FROM hoeren h WHERE h.MatrNr = s.MatrNr) as Anza
    (SELECT COUNT(p.MatrNr) FROM pruefen p WHERE p.MatrNr = s.MatrNr) as Anz
    (SELECT AVG(p.Note) FROM pruefen p WHERE p.MatrNr = s.MatrNr) as Durchsc
FROM Studenten s;
```

```
 * sqlite:///uni.db
Done.
```

Out[33]:

MatrNr	Name	Semester	AnzahlVL	AnzahPR	DurchschnittPR
24002	Xenokrates	18	0	0	None
25403	Jonas	12	1	2	3.0
26120	Fichte	10	1	0	None
26830	Aristoxenos	8	0	2	1.5
27550	Schopenhauer	6	2	1	2.0
28106	Carnap	3	4	3	2.6666666666666665
29120	Theophrastos	2	3	0	None
29555	Feuerbach	2	2	2	1.0

In [42]:

```
%%sql
WITH pruefungen(MatrNr, AnzahlPR, DurchschnittPr) AS (
    SELECT
        MatrNr,
        COUNT(*),
        AVG(Note)
    FROM pruefen
    GROUP BY MatrNr
), hoerend(MatrNr, AnzahlVL) AS (
    SELECT
        MatrNr,
        COUNT(*)
    FROM hoeren
    GROUP BY MatrNr
)
SELECT s.*,
    IFNULL(p.AnzahlPr, 0) AS  AnzahlPr,
    IFNULL(p.DurchschnittPr, 0) AS DurchschnittPr,
    IFNULL(h.AnzahlVL, 0) as AnzahlVL
FROM Studenten s
LEFT JOIN pruefungen p USING(MatrNr)
LEFT JOIN hoerend h USING(MatrNr);
```

```
 * sqlite:///uni.db
Done.
```

Out[42]:

MatrNr	Name	Semester	AnzahlPr	DurchschnittPr	AnzahlVL
24002	Xenokrates	18	0	0	0
25403	Jonas	12	2	3.0	1
26120	Fichte	10	0	0	1
26830	Aristoxenos	8	2	1.5	0
27550	Schopenhauer	6	1	2.0	2
28106	Carnap	3	3	2.6666666666666665	4
29120	Theophrastos	2	0	0	3
29555	Feuerbach	2	2	1.0	2

Aufgabe 6

Wie viele Professoren heißen was mit "krates" am Ende?

In [13]:
```
%%sql

SELECT COUNT(*) AS anzahl FROM Professoren WHERE name LIKE "%krates";
```

```
 * sqlite:///uni.db
Done.
```

Out[13]:

anzahl
1

Aufgabe 7

Welche Studenten wurden von Sokrates geprüft?

In [18]:
```
%%sql
WITH sokrates(MatrNr) AS (
    SELECT MatrNr
    FROM pruefen p
    JOIN Professoren prof USING(PersNr)
    WHERE prof.name = "Sokrates"
)
SELECT s.*
FROM Studenten s
WHERE MatrNr IN ( SELECT MatrNr FROM sokrates )
```

```
 * sqlite:///uni.db
Done.
```

Out[18]:

MatrNr	Name	Semester
25403	Jonas	12
26830	Aristoxenos	8
28106	Carnap	3
29555	Feuerbach	2

In [20]:
```
%%sql
WITH sokrates(MatrNr) AS (
    SELECT MatrNr
    FROM pruefen p
    JOIN Professoren prof USING(PersNr)
    WHERE prof.name = "Sokrates"
)
SELECT DISTINCT s.*
FROM Studenten s
JOIN sokrates so USING(MatrNr);
```

```
 * sqlite:///uni.db
Done.
```

Out[20]:

MatrNr	Name	Semester
25403	Jonas	12
26830	Aristoxenos	8
28106	Carnap	3
29555	Feuerbach	2

In [16]:
```
%%sql

SELECT *
FROM pruefen p
JOIN Professoren prof USING(PersNr)
WHERE prof.name = "Sokrates"
```

```
 * sqlite:///uni.db
Done.
```

Out[16]:

MatrNr	VorlNr	PersNr	Note	Name	Rang	Raum
25403	5041	2125	2	Sokrates	C4	226
25403	4052	2125	4	Sokrates	C4	226
26830	4052	2125	2	Sokrates	C4	226
28106	4052	2125	3	Sokrates	C4	226
29555	4052	2125	1	Sokrates	C4	226
26830	5041	2125	1	Sokrates	C4	226
28106	5041	2125	4	Sokrates	C4	226
29555	5041	2125	1	Sokrates	C4	226

Aufgabe 8

Welche Studenten hören weder eine Vorlesung, noch haben eine Prüfung abgelegt?

In [43]:
```
%%sql
WITH pruefungen(MatrNr, AnzahlPR, DurchschnittPr) AS (
    SELECT
        MatrNr,
        COUNT(*),
        AVG(Note)
    FROM pruefen
    GROUP BY MatrNr
), hoerend(MatrNr, AnzahlVL) AS (
    SELECT
        MatrNr,
        COUNT(*)
    FROM hoeren
    GROUP BY MatrNr
)
SELECT s.*
FROM Studenten s
LEFT JOIN pruefungen p USING(MatrNr)
LEFT JOIN hoerend h USING(MatrNr)
WHERE p.AnzahlPr IS NULL AND h.AnzahlVL IS NULL;
```

```
 * sqlite:///uni.db
Done.
```

Out[43]:

MatrNr	Name	Semester
24002	Xenokrates	18

In [44]:

```
%%sql
WITH pruefungen(MatrNr, AnzahlPR, DurchschnittPr) AS (
    SELECT
        MatrNr,
        COUNT(*),
        AVG(Note)
    FROM pruefen
    GROUP BY MatrNr
), hoerend(MatrNr, AnzahlVL) AS (
    SELECT
        MatrNr,
        COUNT(*)
    FROM hoeren
    GROUP BY MatrNr
), tmp AS (
    SELECT s.*,
        IFNULL(p.AnzahlPr, 0) AS  AnzahlPr,
        IFNULL(p.DurchschnittPr, 0) AS DurchschnittPr,
        IFNULL(h.AnzahlVL, 0) as AnzahlVL
    FROM Studenten s
    LEFT JOIN pruefungen p USING(MatrNr)
    LEFT JOIN hoerend h USING(MatrNr)
)
SELECT * FROM tmp WHERE AnzahlPr = 0 AND AnzahlVL = 0;
```

```
 * sqlite:///uni.db
Done.
```

Out[44]:

MatrNr	Name	Semester	AnzahlPr	DurchschnittPr	AnzahlVL
24002	Xenokrates	18	0	0	0

Aufgabe 10

Welche Stundenten haben noch nie einen Kurs von Sokratest belegt?

In [50]:

```
%%sql

WITH pr_sokrates(MatrNr) AS (
    SELECT p.MatrNr
    FROM pruefen p
    JOIN Professoren prof USING(PersNr)
    WHERE prof.name = "Sokrates"
), ho_sokrates(MatrNr) AS (
    SELECT h.MatrNr
    FROM hoeren h
    JOIN Vorlesungen v USING(VorlNr)
    JOIN Professoren prof ON(prof.PersNr = v.gelesenVon)
    WHERE prof.name = "Sokrates"
), sokrates(MatrNr) AS (
    SELECT MatrNr
    FROM pr_sokrates
    UNION
```

```
    SELECT MatrNr
    FROM ho_sokrates
)
SELECT * FROM Studenten s WHERE s.MatrNr NOT IN (SELECT * FROM sokrates);
```

```
 * sqlite:///uni.db
Done.
```

Out[50]:

MatrNr	Name	Semester
24002	Xenokrates	18
26120	Fichte	10

Aufgabe 11

Suche Studenten (id, name, durchschnittliche Punktzahl), die zwei oder mehr Kurse nicht bestanden haben

oder:

Aufgabe 12

Welche Studenten sind in Ethik durchgefallen? Sortieren Sie nach Matirkelnummer absteigend.

In [53]:

```
%%sql

SELECT *
FROM pruefen p
JOIN Vorlesungen v USING(VorlNr)
WHERE v.Titel = "Ethik" AND Note > 4
ORDER BY p.MatrNr DESC
```

```
 * sqlite:///uni.db
Done.
```

Out[53]:

MatrNr	VorlNr	PersNr	Note	Titel	SWS	gelesenVon

Aufgabe 13

Zeige für jeden Studenten die Noten aller Kurse im Vergleich zur durchschnittlich erreichten Punktzahl in diesem Kurs

In [61]:

```
%%sql

WITH durchschnitt(VorlNr, Note) AS (
    SELECT VorlNr, AVG(Note)
    FROM pruefen
    GROUP BY VorlNr)
SELECT s.*, v.Titel, p.Note, d.Note as durchschnitt
FROM Studenten s
JOIN pruefen p USING(MatrNr)
JOIN Vorlesungen v USING(VorlNr)
```

```
JOIN durchschnitt d USING(VorlNr)
ORDER BY s.MatrNr;
```

```
 * sqlite:///uni.db
Done.
```

Out[61]:

MatrNr	Name	Semester	Titel	Note	durchschnitt
25403	Jonas	12	Logik	4	2.5
25403	Jonas	12	Ethik	2	2.0
26830	Aristoxenos	8	Logik	2	2.5
26830	Aristoxenos	8	Ethik	1	2.0
27550	Schopenhauer	6	Die 3 Kritiken	2	2.0
28106	Carnap	3	Logik	3	2.5
28106	Carnap	3	Grundzuege	1	1.0
28106	Carnap	3	Ethik	4	2.0
29555	Feuerbach	2	Logik	1	2.5
29555	Feuerbach	2	Ethik	1	2.0

Aufgabe 14

Beste, Schlechteste und Durchschnittsnote pro Kurs

Aufgabe 15

Erstellen Sie ein Ranking der besten Studenten pro Kurs. Geben Sie den Namen des Kurses, den Namen des Studenten, dessen Note und dessen Ranking an.

Aufgabe 16

Finden Sie die beiden besten Studenten pro Vorlesung