Exercise 1

Answer the following questions on our university database using SQL:

(a) Which Professors does Fichte know from attending their Lectures.

(b) Which Lectures are attended by Students in the 1.-4. semester? Print only the title of the lectures.

(c) Find all Students that attend at least one Lecture together with Fichte.

Lösung:

(a) Which Professors does Fichte know from attending their Lectures.

```
select distinct p.persNr, p.name 
from Professors p, attend a, Lectures l, Students s 
where p.PersNr = l.given_by 
and l.lectureNr = a.lectureNr 
and a.stud Nr = s.studNr 
and s.name = 'Fichte';
```

(b) Which Lectures are attended by Students in the 1.-4. semester? Print only the title of the lectures.

```
select distinct l.title 
from Lectures l, attend a, Students s 
where l.lectureNr = a.lectureNr 
and a.studNr = s.studnr 
and s.semester between 1 and 4;
```

(c) Find all Students that attend at least one Lecture together with Fichte.

```
select distinct other_s.studNr, other_s.name 
from Students fichte_s, attend fichte_a, attend other_a, Students other_s 
where fichte_s.name = 'Fichte' 
and fichte_a.studNr = fichte_s.studNr 
and other_a.lectureNr = fichte_a.lectureNr 
and other_s.studNr = other_a.studNr 
and other_s.studNr <> fichte_s.studNr
```

Exercise 2

Answer the following questions on our university database using SQL:

a) Figure out the average semester of the all students.
b) Determine the average semester of students that attend at least one lecture of Sokrates.

c) Calculate how many lectures students are attending on average. Students who do not attend any lecture should be reflected in the result as well. If you get stuck, see hints: 

d) Calculate how many lectures each student is attending. Students who do not attend any lecture should be included in the result as well (attend_count = 0).

Solution:

a) Figure out the average semester of the all students.

```sql
select avg(semester) from students;
```

b) Determine the average semester of students that attend at least one lecture of Sokrates.

```sql
select avg(semester)
from students s
where exists (  
    select *
    from attend a, lectures l, professors p
    where s.studnr = a.studnr  
    and a.lecturenr = l.lecturenr  
    and l.given_by = p.persnr  
    and p.name = 'Sokrates'
)
```

In this query we need to make sure that each student is only counted once, even if she is attending two lectures by Sokrates. In our solution, the use of `exists` takes care of this. However, we could have also used `distinct` in combination with a sub-query:

```sql
select avg(semester)
from (select distinct s.*
    from Students s, attend a, lectures l, professors p
    where s.studnr = a.studnr  
    and a.lecturenr = l.lecturenr  
    and l.given_by = p.persnr  
    and p.name = 'Sokrates')
```

---

1 Remember that the from clause is optional (`select 1.0 / 2.0;` is a valid query).
2 Remember that you can use sub-queries in the select clause.
c) Calculate how many lectures students are attending on average. Students who do not attend any lecture should be reflected in the result as well.

```sql
select attend_count / (student_count * 1.000)
from (select count(*) as attend_count from attend) a,
( select count(*) as student_count from students ) s
```

Or:

```sql
select attend_count / cast (student_count as numeric (10,4))
from (select count(*) as attend_count from attend) a,
( select count(*) as student_count from students ) s
```

d) [Bonus] Calculate how many lectures each student is attending. Students who do not attend any lecture should be included in the result as well (attend_count = 0).

In this exercise we have to make sure to include students that do not attend any lecture.

```sql
select s.studnr, s.name, (select count(*) from attend a where a.studnr = s.studnr)
from students s;
```

Another possible solution would be to use `union`. We first calculate the number of attended lectures for each student that does attend a lecture. Then we create a query that produces the student number, student name and a zero for all students that do not attend a lecture. We then simply combine the two results using the `union` operator. Note, however, that it is important to only allow students that do not attend any lecture in the second sub-query. Otherwise, duplicates would be possible.

```sql
(select s.studnr, s.name, count(*)
from students s, attend a
where s.studnr = a.studnr
group by s.studnr, s.name)
union
(select s1.studnr, s1.name, 0 as cnt
from students s1
where not exists (select * from attend a where a.studnr = s1.studnr))
```

A similar approach that takes care of duplicates in a different way is shown in the following query. Here we do not avoid duplicates, but filter them out in a second step, instead.

```sql
select x.studnr, x.name, sum(x.cnt)
from
((
select s2.studnr, s2.name, count(*) as cnt
from students s2, attend a
where s2.studnr = a.studnr
group by s2.studnr, s2.name)
union
(select s1.studnr, s1.name, 0 as cnt
from students s1
)) x
```
As should be clear from this exercise, there are many different ways how a query can be written. As a rule of thumb, shorter queries are often better, because these are easier to understand. That holds for everyone involved: you yourself (when proofreading your queries in the exam), other people (who read your queries and need to understand them) and the database (which has to execute your queries in an efficient manner).